ENERGY-AWARE RESOURCE ALLOCATION FOR EFFICIENT MANAGEMENT OF DATA CENTERS FOR CLOUD COMPUTING

BY

Ali Raza

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

December 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by

ALI RAZA

under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

Thesis Committee

Dr. Sadiq M. Sait
(Adviser)

Dr. Alaaeldin Amin
(Member)

Dr. Ashraf S. Hasan Mahmoud
(Member)

Dr. Ahmad Almulhem
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

13/1/16
Date

To my parents, whose words of encouragement and push for tenacity ring in my ears. To my brothers and sisters, who have supported me throughout the process.

ACKNOWLEDGMENTS

First of all, infinite thanks and complete gratitude to the almighty Allah for his unstoppable blessings. Many thanks, to my father, my mother, and my whole family for their unwavering trust and support.

Special thanks to my mentor and advisor Dr. Sadiq M. Sait for his appreciated guidance and wisdom. Thanks to my committee members: Dr. Alaaeldin Amin and Dr. Ashraf S. Hasan Mahmoud for their motivation and valuable suggestions. I would especially like to thank Khawaja Shahzada for being a great research partner, and helping me through my masters in more ways than I can remember. I would also like to thank Abubakar Bala for his valuable contributions.

Thanks to COE department and KFUPM for giving me this opportunity. Special thanks are due to my senior colleagues at KFUPM who supported me with help and encouragement during the work. Finally, thanks to everybody who contributed to this achievement in a direct or an indirect way.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VM	Virtual Machine
PM	Physical Machine
VMP	Virtual Machine Placement Problem
VBP	Vector Bin Packing Problem
CRAC	Computer Room AC
SE	Simulated Annealing
TS	Tabu Search
SimE	Simulation Annealing
FFD	First Fit Decreasing
WFD	Worst Fit Decreasing
LL	Least Loaded
Saas	Software as a Service
Paas	Platform as a service
IaaS	Infrastructure as a Service
COP	Coefficient of Performance

THESIS ABSTRACT

NAME: Ali Raza

TITLE OF STUDY: Energy-aware resource allocation for efficient management of data centers for cloud computing

MAJOR FIELD: Computer Engineering

DATE OF DEGREE: December, 2015

Cloud computing has evolved as the next-generation platform for hosting applications ranging from sciences to business, and social networking to media content delivery. The numerous data centers, employed to provide cloud services consume large amounts of power, both for their functioning and their cooling. Improving power efficiency, that is, decreasing the total power consumed, has become an important task for many data centers for reasons such as cost, infrastructural limits, and negative environmental impact. Power management is a challenging optimization problem due to the scale of modern data centers and presence of conflicting objectives. Most published work focuses on power management in computing nodes and that in the cooling facility in an isolated manner. In this work, we use a combination of spatial subsetting and thermal management to optimize the

total power consumed by the computing nodes and the cooling facility; and show that solutions, that do not consider these interactions and are not holistic, do not necessarily result in minimum total power consumption. We employ three evolutionary non-deterministic heuristics; namely, Simulated Annealing(SA), Tabu Search (TS), and Simulated Evolution (SimE) to find the best location of each virtual machine (VM) to a physical machine in a data center, based on its computational power and data center heat recirculation model, to optimize the total power consumption. Experimental results for wide range of different problem instances show that proposed approach outperforms heuristics proposed in previous studies.

ABSTRACT (ARABIC)

ملخص الرسالة

الاسم الكامل: على رضا

عنوان الرسالة: تخصيص الموارد بكفاءة الطاقة للادارة الفعالة لمراكز بيانات الحوسبة السحابية

التخصص: هندسة حاسب آلي

تاريخ الدرجة العلميه: ديسمبر ٢٠١٥

لقد تطورت الحوسبة السحابية باعتبارها منصة الجيل القادم لاستضافة التطبيقات التي تتراوح بين العلوم والأعمال، والشبكات الاجتماعية وتقديم محتوى وسائل الإعلام. راكز البيانات العديدة، التي استخدمت لتقديم الخدمات السحابية تستهلك كميات كبيرة من الطاقة، سواء للعمل أو التبريد الخاصة بها. لذا فإن تحسين كفاءة الطاقة والحد من إجمالي الطاقة المستهلكة، أصبحت مهمة هامة للعديد من مراكز البيانات لأسباب مثل التكلفة وحدود البنية التحتية، والآثار البيئية السلبية. إدارة الطاقة هي الحل الأمثل لهذه المشكلة نظرا لحجم مراكز البيانات الحديثة ووجود عدة أهداف متعارضة. وتركز معظم الأعمال المنشورة على إدارة الطاقة في عقد الحوسبة وكذلك في مرافق التبريد بطريقة منفصلة. في هذا العمل، نستخدم مزيج من الحيز المكاني والإدارة الحرارية لتحسين إجمالي الطاقة المستهلكة من قبل عقد الحوسبة ومرافق التبريد. ونبين أن الحلول، التي تفصل بين الجهتين ليست شاملة، ولا تؤدي بالضرورة إلى أدنى استهلاك للطاقة. تم توظيف ثلاثة خوارزميات تطورية غير قطعية. وهي محاكي التلايين (SA)، البحث الممنوع (TS)، ومحاكي التطور (SimE) للعثور على أفضل مكان لكل جهاز افتراضي (VM) في مركز البيانات، على أساس القدرة الحسابية ونموذج إعادة التدوير الحراري لمركز البيانات ، لتحسين استهلاك الطاقة الإجمالية. النتائج التجريبية لمجموعة واسعة من الحالات المختلفة تبين أن النهج المقترح يتفوق الأساليب البحثية المقترحة في الدراسات السابقة.

CHAPTER 1

INTRODUCTION

Over the past decade, cloud-based data centers have emerged as a popular computing paradigm. Due to computational capabilities and rapid elasticity, the IT industry is rapidly adopting cloud computing for hosting and delivering Internet based applications and services [2]. As a result of such a proliferation, there has been an increase in the power density and power consumption of data centers. In 2013, 91 billion kWh of power was consumed by U.S. data centers. Data center electricity consumption is estimated to increase to approximately 140 billion kWh annually by 2020 which is equivalent to $13 billion per year in electricity bills, and emission of nearly 150 million metric tons of annual carbon pollution [3].

In some cases, power supply available for data centers is limited. For example, Morgan Stanley was not able to run a new data center in Manhattan due to unavailability of the power needed to operate it [4]. Power availability has been identified as a key factor by 30% of data center providers for limiting server deployment [5]. Power required by the computing nodes and the cooling facility

in a data center is a crucial issue for data center providers since it dominates the operational costs. For example, based on a 3-year amortization schedule, Amazon estimated that cost and operation of the severs at its data centers accounts for 53% of the total budget, cost related to power is 42% of the total including infrastructure for cooling and direct power consumption [6]. Therefore, considerable power budget for operating data centers can be saved and significant contribution can be made to greater environmental sustainability by improving power efficiency in cloud data centers.

A typical data center hosts number of servers corresponding to maximum load, but the utilization invariably is around 30-70% [7][8]. Significant percentage of servers sit idle waiting for user requests and consume about 60-70% of full utilization power [9][10][11]. This significant idle power not only adds to the computational power consumed but also to that consumed by the cooling facility to remove heat. *Spatial subsetting* has been proposed to reduce idle power by assigning user requests to a subset of servers and powering-off the rest of the servers [12][13]. It is ensured that the active servers are sufficient to meet the computational demand based on current loading. More servers are powered-on as more user requests arrive. *Spatial subsetting* attempts to concentrate user requests on minimum number of servers that results in high utilization per server. The servers consume large amount of computational power because of higher utilization, and this power is dissipated as heat. Consequently, the active servers reach higher temperature due to the hot spots creation by the heat dissipated.

This heat needs to be extracted to avoid overheating and failing of servers [14]. Computer Room AC (CRAC) units provide cold air, which enters the servers through the front air inlets in the cold aisle, picks up heat from the circuitry, and exits via outlet to the hot aisles. Accordingly, the outlet temperature rises compared to the inlet temperature. Air conditioners positioned above the hot aisle extract this hot air, but a large fraction of this heat recirculates to the cold aisle increasing the inlet temperature. The temperature of supplied air is adjusted (reduced) so that the inlet temperature is lower than the safe value required to avoid over heating of servers. Cooling facilities exert more effort to provide cold air at lower temperature, i.e., more electric power is required by the cooling facility at higher inlet temperature. Techniques have been proposed to optimize the maximum temperature in data centers to minimize power consumption by the cooling facility [15] [16] [17] [18]. Requests are distributed among all the servers to maintain a uniform reduced temperature. Due to data centers' configuration, servers under the same utilization can reach different temperature. For example, air flow is better near the AC outlets as compared to far ends of the servers' racks. Every request is assigned to the server with minimum temperature, servers' temperature is recalculated and the next request is considered for assignment. Thus requests are distributed among all the servers to minimize the maximum temperature. This strategy is known as inverse-temperature assignment [19]. This technique optimizes the maximum temperature to reduce cooling cost, but it does so by distributing user requests among all servers therefore increases idle power.

Thus, inverse-temperature assignment can increase idle power more than it reduces cooling power, and vice versa for *spatial subsetting*. There is a trade-off between cooling power and computational power, and both should be optimized together to minimize the total power consumption in data centers.

1.1 Significance of the problem

Given the hardware of already established data centers cannot be improved, there are two main ways to reduce server power: 1) dynamic voltage and frequency scaling (DVFS); 2) VM consolidation. The first technique can be enabled on the firmware level, or on the hypervisor level as the local strategy, and there is no more need to regulate it in the global level algorithm. However, this technique can save only a small percentage of energy based on the current hardware design. According to the test results in the manual from HP [20], the saved energy in this way on the types of servers indicated is about 5%. The second technology can save a large amount of energy, but it is more complex. Assigning VMs to fewer servers and turning off others can save power because the addition of a new server to the system usually implies some fixed power-consumption increase [21][22], even if such server is not fully loaded. Assigning VMs to a cluster of minimum servers can lead to hot spots, due to higher utilization per server, that can be costly in terms of power required by the cooling infrastructure. Significant energy can be saved if cooling power and computing power are minimized in an integrated fashion. Hence, the energy conservation technique based on VM placement should

be researched further.

1.2 Research Objective

The aim of this work is to investigate the research challenges in relation to energy-efficient VM consolidation in data centers. In particular, the interaction between server consolidation and thermal management is explored, and an integrated solution to VM allocation problem is presented. Efficient evolutionary non-deterministic heuristics are employed to reduce operational cost by reducing total power consumption in data centers.

1.3 Thesis Contribution

Contribution of this work can be summarized as follows:

- Since minimizing number of servers doesn't reduce the total power consumed in data centers, problem is formulated to minimize the total power required by the computing nodes and the cooling facility by considering the trade-off between VM consolation and the thermal model of data centers.
- Simulated Annealing and Tabu Search are employed for integrated optimization of computational power and cooling power.
- Simulated Evolution is engineered for joint optimization of the power required by the computing nodes and the cooling facility. A goodness measure is proposed to minimize the number the servers and heat recirculation in an

integrated fashion.

- Performance of the proposed method is compared with well-known algorithms for a wide range of different problem instances

1.4 Thesis Organization

The remainder of this thesis is organized as follows: Chapter 2 discusses background of the problem and related work in this area. Chapter 3 describes data center configuration and power model and formally defines the problem. Chapter 4 briefly explains iterative non-deterministic heuristics. Experimental methodology and results are reported in Chapter 5. Chapter 6 concludes this work and suggests future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Cloud computing

Cloud computing generally refers to the deployment and usage of compute and storage resources over the internet. The usage of these resources follow pay-as-you-go pricing models. The most comprehensive definition of cloud computing, provided by the National Institute of Standards and Technology (NIST) is as follows "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [23]."

2.2 Deployment Models

The word "cloud" refers to a big network of shared resources. The cloud infrastructure may be designed either to serve different functional units of the single organization or to be shared among several organizations to reduce the overall infrastructure and operational cost of the IT sector. From the deployment point of view, the cloud infrastructure has four major types: *Private*, *Public*, *Community* and *Hybrid cloud*. In a private cloud, the cloud infrastructure is solely utilized by a single organization, while in public cloud, several organizations can publicly use these resources on lease basis through a network link. Community cloud also shares resources with multiple organizations, however it slightly differs from public cloud in a sense that provision of resources is restricted to group of organizations that share common concerns and belong to a specific community.

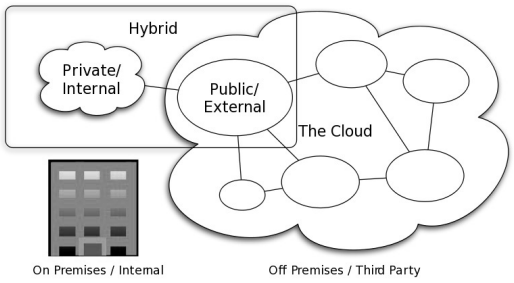


Figure 2.1: Cloud Deployment Models [1].

2.3 Service Models

Architecture of a cloud computing environment is best described by a layered model as shown in Fig. 2.2. This model has three layers namely *Application*,

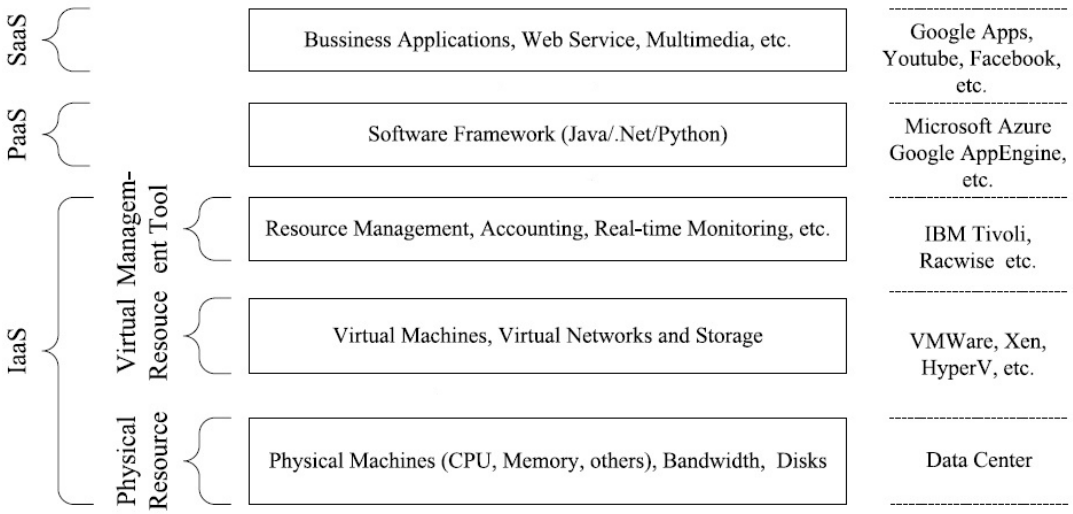


Figure 2.2: Cloud service models.

Platform and *Infrastructure*. Each one of these layers provides a different type of services. At the application layer, all the underlying hardware and operating system information is hidden from the end users. Users can access the cloud hosted applications through a web browser. This type of service is known as Software-as-a-service (SaaS). Facebook.com and salesforce.com are common examples of this service model. At the second layer, cloud provider offers computing platforms such as operating systems and application frameworks. At this level, users have the privileges to deploy their own applications. Microsoft Azure, Google App Engine and Amazon SimpleDB/S3 are typical examples [24]. This service model is known as Platform-as-a-service (PaaS). Lowest layer in the hierarchy offers Infrastructure-as-a-service (IaaS). It manages fundamental computing resources that include storage, processing, network, etc. These resources are provided to the end user, usually in terms of storage blocks and virtual machines (VMs) using virtualization technologies such as VMware [25], KVM [26], and Xen [27]. This

layer is also known as virtualization layer [24].

2.4 Virtualization

Virtualization is one of the prominent technologies that makes cloud computing possible. It allows resources of a single large server to be sliced into multiple isolated execution environments so that multiple operating systems can co-exist on a single physical machine. User requests are translated into computational requirements that are mapped to VMs with desired characteristics. Multiple VMs are assigned to a single physical server, sharing the same underlying machine's computing resources, which results in fewer physical servers. Fig. 2.3 illustrates the benefits of virtualization and how optimal assignment helps to minimize the number of active servers.

Five applications are running on five different servers. This is a wastage of resources since all of the servers are underutilized. With virtualization, we can translate these applications into VMs, and these VMs can run on fewer number of servers as illustrated in Fig. 2.4.

Virtualization gives immense benefits in terms of higher utilization per server, increased flexibility and availability, reducing hardware costs and physical space, etc. However, cost of this flexibility is that a pool of VMs with user-defined specifications must be implemented by, or assigned to various physical machines (servers), a problem known as Virtual Machine Placement [28]. Due to multi-dimensional nature of VM requests, the assignment problem is very challenging.

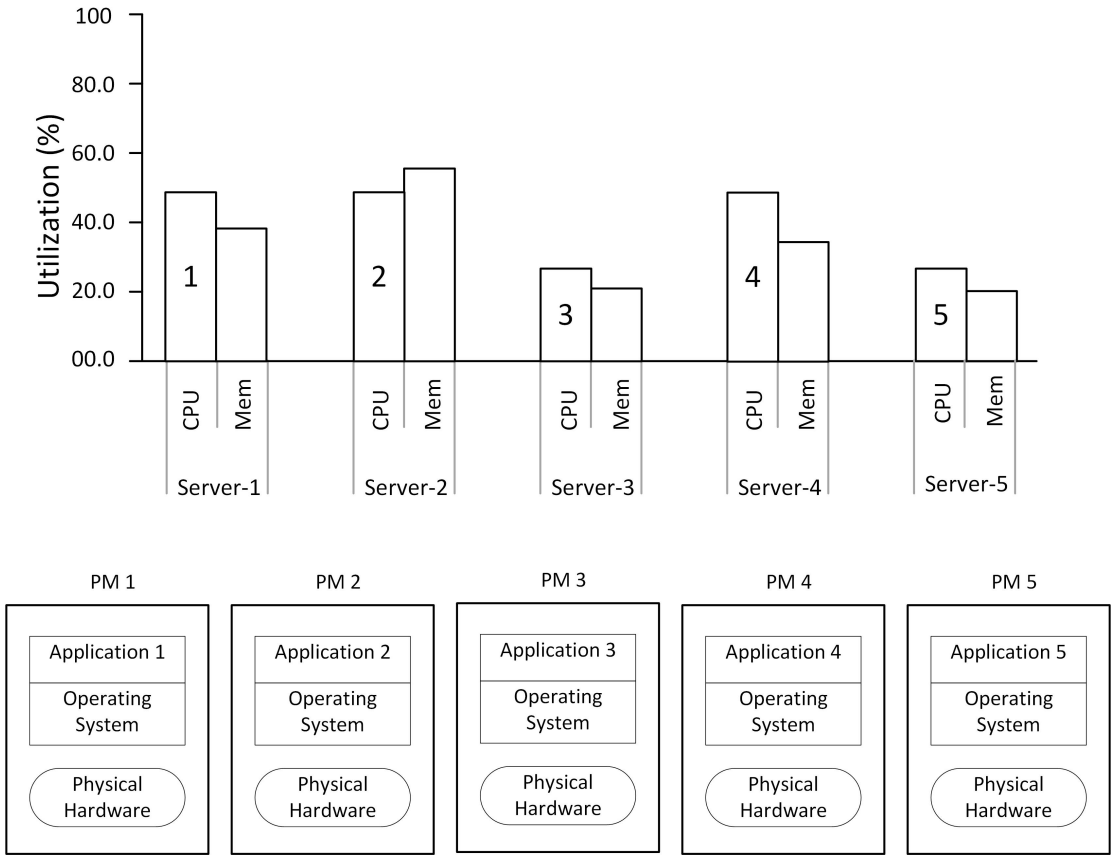


Figure 2.3: Applications running on individual servers.

Every VM has its own CPU, memory, and bandwidth requirements. Similarly, every server has a fixed capacity across each of these dimensions. VM to server assignment should not violate these capacity constraints. VM assignment is often formulated as a vector bin packing problem (VBP) [29] [30], where the VMs that are treated as objects (n) are packed into servers that are treated as bins (b). The computational complexity of VBP is $O(b^n)$. Clearly, it is impractical to enumerate all possible assignment for a large number of VMs (objects). Even the one-dimensional version of this problem is NP-hard. Finding optimal assignment to such problems is computationally infeasible for large problem sets. Heuristics can be used to find near-optimal assignment in polynomial time.

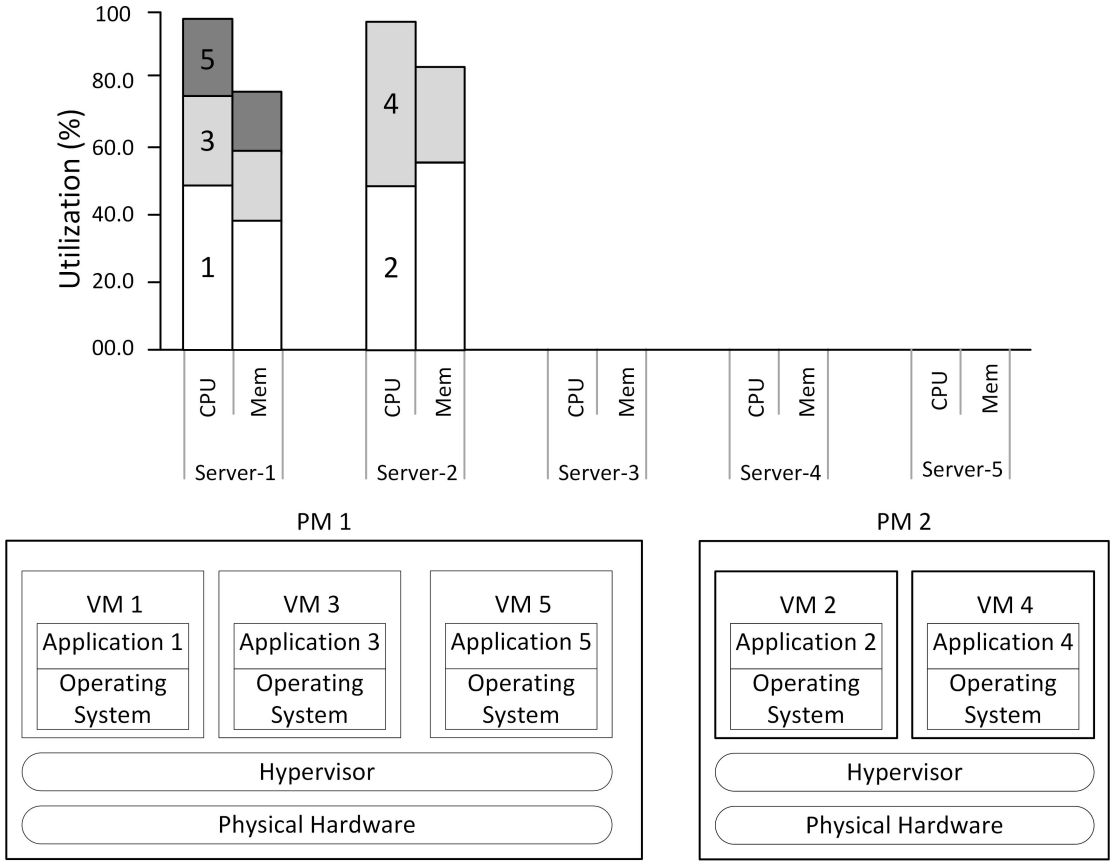


Figure 2.4: Applications running on individual VMs that share physical resources.

2.5 Related Work

Computation nodes (servers) account for 30% of total power consumption in data center [2] and in the last few years, much work has been done for optimal VM assignment with the objective of minimizing number of active servers (PMs), thereby decreasing computational power. Gao *et al.* [28] proposed a modified ant colony optimization algorithm that minimizes total resource wastage and power consumption in physical machines. Karmer *et al.* [31] used VM consolidation with the concept of dynamic voltage/frequency scaling (DVFS) to improve the power efficiency. Doddvula *et al.* [32] proposed a Magnitude Classified algorithm based

on First Fit Decreasing (FFD) for server consolidation. Ajiro *et al.* [33] suggested improvements to the classical FFD and least loaded (LL) algorithms to optimize computational power. All of these schemes try to minimize computational power by minimizing active severs and ignore cooling infrastructure despite it causing a significant fraction of total power.

Cooling facility is used to remove the heat dissipated by the computing nodes. To minimize the power consumption of cooling infrastructure, Sullivan *et al.* [34] and Patel *et al.* [35] optimized cooling power by improving air flow. Moore *et al.* [17] proposed to predict heat profiles using software infrastructure. Mukherjee *et al.* [36] used a software based infrastructure to control resource management for thermal-aware allocation. These schemes only optimize cooling power.

Al-Qawasmeh, A., *et al.* [37] used nonlinear programming technique to optimize cooling and idle power. Non-linear programming technique has poor scalability, and it may not be suitable for practical scenarios.

CHAPTER 3

PROBLEM DESCRIPTION

This chapter provides the necessary preliminaries and system models that are required to define the problem. Table 3.1 shows the parameters and notations used throughout this thesis.

3.1 A Typical Data Center Configuration

A typical data center is arranged in a hot-aisle/cold-aisle configuration as shown in Fig. 3.1. Racks containing servers are installed on a raised floor. All servers are connected to a high-speed network, typically in star topology over central switch. CRAC units extract hot air from the top and deliver cold air through pressurized floor plenum. The pressure forces the cold air upward through the perforated floor tiles. Power consumed by the servers is dissipated as heat. The cold air enters the servers through front air inlets in the cold aisle, picks up heat from the circuitry, and exits to the hot aisles via outlets. Air conditioners positioned above the hot aisle extract this hot air.

Table 3.1: Notations and definitions.

Symbol	Definition
VM	virtual machine
PM	physical machine
n	number of VMs
m	number of servers(PMs)
P_{total}	total power
P_{AC}	cooling power
P_{CN}	computational power
P_j^{idle}	idle computational power of j^{th} server
P_j^{busy}	average computational power of j^{th} server in 100% utilization
U_j^p	CPU utilization of j^{th} server
U_j^m	memory utilization of j^{th} server
x_{ij}	1 if i^{th} VM is assigned to j^{th} server otherwise 0
y_j	1 if j^{th} server is ON otherwise 0
v_i^c	CPU requirement of i^{th} VM
v_i^m	memory requirement of i^{th} VM
COP	coefficient of performance
T_{sup}	temperature of air supplied by CRAC
T_{red}	Manufacturer-specified maximum inlet temperature typical value 25^oC
ρ	air density in kg/m^3
f	air flow rate in m^3/s
Q	heat rate in Watt (W)
c_p	specific heat of air in $kJ/kg.Kelvin$
$\vec{A}$	heat cross-interference coefficient matrix
T_{out}^i	inlet temperature of i^{th} server
T_{in}^i	outlet temperature of i^{th} server
$\vec{K}$	thermodynamic constant matrix, $\vec{K} = \text{diag}(K_i)$
$\overrightarrow{T_{in}}$	the vector $\{T_{in}^i\}_n$
$\overrightarrow{T_{out}}$	the vector $\{T_{out}^i\}_n$
$\vec{D}$	distribution matrix, concise for $[(\vec{K} - \vec{A}^T\vec{K})^{-1} - \vec{K}^{-1}]$

3.2 Power Consumption in Data Centers

We define total power (P_{total}) of a data center as a sum of computational power (P_{CN}) and cooling power (P_{AC}). We are not considering power consumed by network devices; storage; lightning; humidifier; and losses due to distribution network consisting of switch gear, conductors, DC-AC and AC-DC converters,

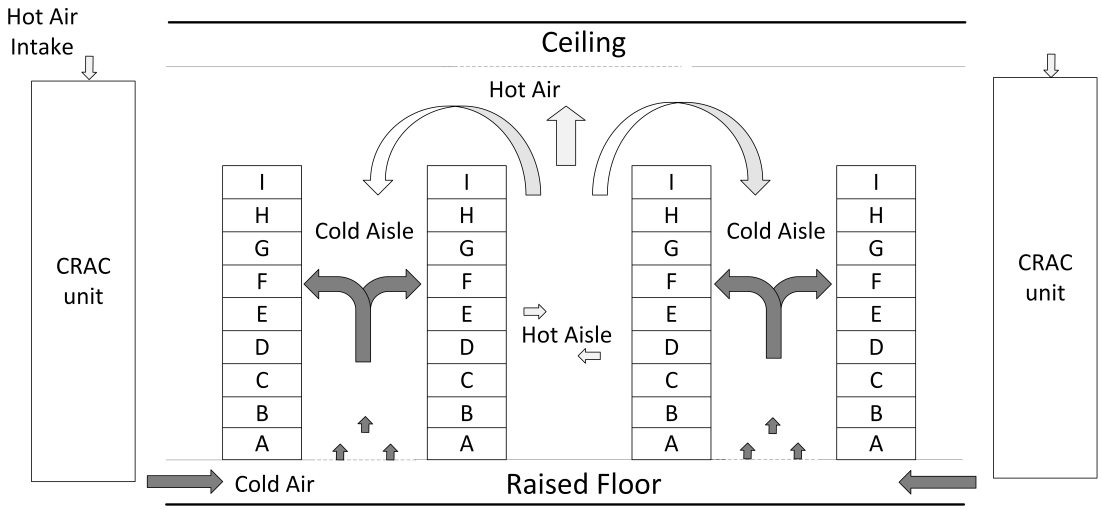


Figure 3.1: A typical data center configuration.

and UPS.

$$P_{total} = P_{CN} + P_{AC} \tag{3.1}$$

3.2.1 Computational Power

Power consumption by computing nodes varies significantly with the actual computing activity. Fan *et al.* [38] conducted experiments to estimate power utilization using performance counters and system activity measurements. They proposed that computational power consumption of servers can be accurately described by a linear model. Their work has been further verified by Gao *et al.* [28] through their experiments conducted on a Dell server. In order to save power, servers that are not being used are turned off. Thus their power consumption in OFF state is not part of the total power consumed by the CPU. Power consumed by j^{th} server

can be defined by Equation 3.2 [28].

$$P_j = \begin{cases} (P_j^{busy} - P_j^{idle}) \times U_j^p + P_j^{idle} : & otherwise \\ 0 : & \text{if } U_j^p = 0 \end{cases} \tag{3.2}$$

where U_j^p is the CPU utilization of j^{th} server, P_j^{busy} is the average power value when j^th server is fully utilized and P_j^{idle} is the average power values when the server is in idle state.

Total power consumption of all computing nodes (P_{CN}) in a data center is calculated as:

$$P_{CN} = \sum_{j=1}^{m} P_j = \sum_{j=1}^{m} \left[y_j \times \left((P_j^{busy} - P_j^{idle}) \times \sum_{i=1}^{n} (x_{ij}.v_i^c) + P_j^{idle} \right) \right] \tag{3.3}$$

where v_i^c is the CPU requirement of i^{th} VM; x_{ij} is the assignment variable, its value is 1 if i^{th} VM is assigned to the j^{th} server otherwise its value is zero; and y_j is 1 if j^{th} server is ON otherwise its value is zero.

3.2.2 Cooling Power

Computer Room AC (CRAC) units are used to remove heat generated by the servers. Performance of a CRAC unit depends on various factors such as construction material and air flow rate [39], and it is quantified by *Coefficient of Performance* (COP). Power consumed by the CRAC units depends on the COP criterion. COP is defined as the ratio of amount of heat removed (Q_s) by the CRAC unit to the total power consumed by cooling facility (P_{AC}) for cooling

process (Equation 3.4) [17].

$$COP(T_{sup}) = \frac{Q_s}{P_{AC}} \tag{3.4}$$

Power consumed by the cooling facility (P_{AC}) is inversely proportional to COP. A higher value of COP, therefore, indicates higher efficiency. Since power consumed by the server (P_{CN}) is dissipated as heat and cooing facility is used to remove this heat, power consumed by the cooling facility can be defined by Equation 3.5 [17].

$$P_{AC} = \frac{P_{CN}}{COP(T_{sup})} \tag{3.5}$$

where P_{CN} is the power consumed by computing nodes.

COP is not constant. It varies with temperature of the air supplied (T_{sup}) by the CRAC units. We are using COP model of chilled-water CRAC units at the HP Labs Utility Data Center [17], which is defined by Equation 3.6.

$$COP(T_{sup}) = 0.0068T_{sup}^2 + 0.0008T_{sup} + 0.458 \tag{3.6}$$

As temperature (T_{sup}) of air supplied by CRAC units air increases, COP increases as shown in Fig. 3.2 and CRAC units use less cooling power (P_{AC}) to remove the same amount of heat (computational power).

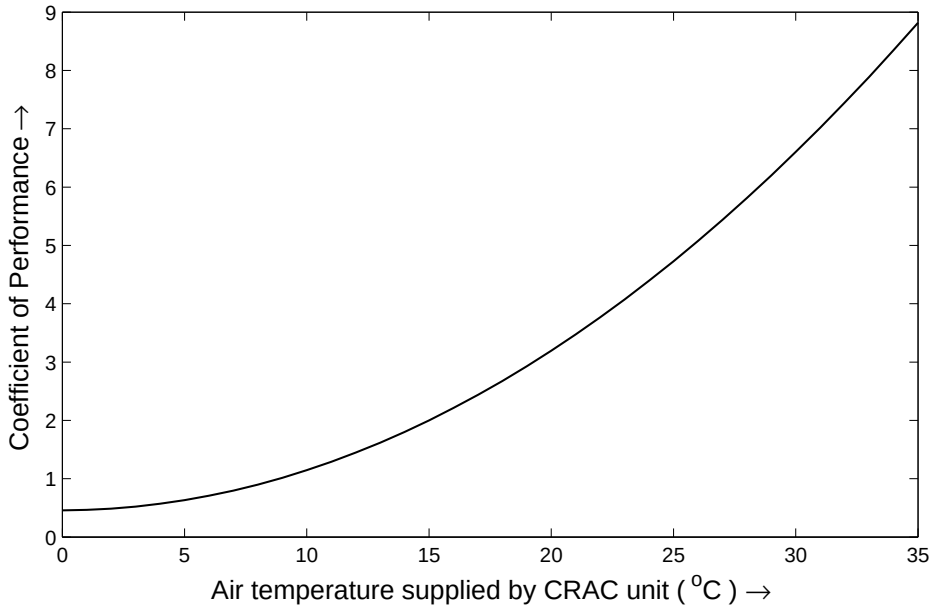


Figure 3.2: Coefficient of performance (COP) curve for the chilled-water CRAC units at the HP Laboratories Utility Data Center.

3.2.3 Total Power Consumption

Using Equations 3.3 and 3.5, total power consumption can be written as:

$$P_{total} = (1 + \frac{1}{COP(T_{sup})})P_{CN} = (1 + \frac{1}{COP(T_{sup})}) \sum_{j=1}^{m} P_j \tag{3.7}$$

3.3 Heat Recirculation in Data Centers

In this section, the effect of heat recirculation in data centers is investigated, and a thermal model based on heat recirculation is discussed which is used to determine the temperature (T_{sup}) of the air supplied by the CRAC units to remove the heat generated by computing nodes. Cooling power (P_{AC}) is computed using T_{sup} as given by Equation 3.7.

According to law of conservation of energy, consumed computational power is

equivalent to heat carried by air per unit time. Amount of heat transferred by air per unit time is calculated as:

$$Q = \rho f c_p T \tag{3.8}$$

Since power consumed by computing nodes is dissipated as heat, it can be written as difference of outgoing hot air temperature (T_{out}) and the supplied cold air temperature (T_{in}):

$$P_i = \triangle Q \Rightarrow P_i = \rho f c_p (T^i_{out} - T^i_{in}) \tag{3.9}$$

Heat recirculation can be described as a phenomenon of server's outlet heat recirculating and affecting the inlet temperature of another server. Tang *et al.* [40] [41] showed that a cross interference matrix, obtained by computational fluid dynamic simulation, can be used to define heat recirculation. Cross interference matrix is denoted as $A_{m \times m}$, where each element a_{ij} is the fraction of heat transferred from the outlet of i^{th} server to the inlet of j^{th} server.

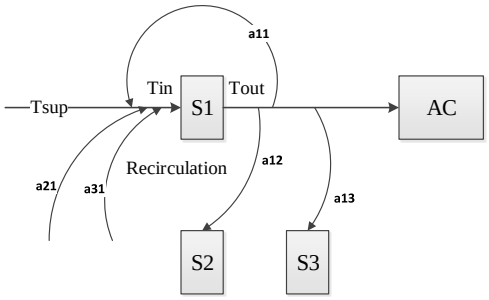


Figure 3.3: Cross interference among computational nodes. Exhaust hot air from server 1 will be partially removed by AC and partially recirculate to other servers' inlets. Whereas, it inhales hot air from other servers.

$$\vec{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & a_{ij} & \vdots \\ a_{m1} & a_{m3} & \dots & a_{mm} \end{bmatrix}$$

$\vec{A}$ essentially captures the physical layout of a data center. The heat rate of inlet air for i^{th} server (Q^i_{in}) is calculated as [41]:

$$Q^i_{in} = \sum_{j=1}^{m} a_{ji} Q^i_{out} + Q_s \tag{3.10}$$

where Q_s denotes heat rate of cold air supplied by CRAC.

We can use thermodynamic constants to transform air heat rate to temperature. Objective is to determine inlet temperature of servers considering heat recirculation, and adjust the temperature of cold air supplied by the CRAC unit to reduce inlet temperature to a safe value to avoid overheating. To this end, we convert matrix $A_{m\times m}$ into the heat distribution matrix $D_{m\times m}$, using Equation 3.12. Assume that $\overrightarrow{T_{in}}$ is the inlet temperature vector and $\overrightarrow{P}$ represents power consumption vector of m servers, and $\overrightarrow{T_s}$ denotes vector containing temperature of supplied air defined as:

$$\begin{pmatrix} T^1_{in} \\ T^2_{in} \\ \vdots \\ T^m_{in} \end{pmatrix}, \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_m \end{pmatrix}, and \begin{pmatrix} T_{sup} \\ T_{sup} \\ \vdots \\ T_{sup} \end{pmatrix}$$

respectively. Similarly, $\vec{K}$ is a $m \times m$ diagonal matrix

$$\vec{K} = \begin{bmatrix} K_1 & 0 & \dots & 0 \\ 0 & K_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & K_m \end{bmatrix}$$

where $K_i = \rho f_i c_p$.

The relation between power consumption (air heat rate) and temperature can be defined similar to [41] as:

$$\overrightarrow{T_{in}} = \overrightarrow{T_s} + \vec{D} \times \overrightarrow{P} \tag{3.11}$$

where

$$\vec{D} = [(\vec{K} - \vec{A}^T \vec{K})^{-1} - \vec{K}^{-1}] \tag{3.12}$$

and

$$T_{in} \leq T_{red} \tag{3.13}$$

Which means that each inlet temperature rises due to heat from recirculation. The temperature of supplied air is adjusted so that the inlet temperature is lower than a specific value (T_{red}). This is necessary to avoid overheating and failing of servers [14]. In this work, we use $T_{red} = 25^oC$ [41]. Equation 3.7 can be rewritten as:

$$P_{total} = (1 + \frac{1}{COP(T'_{sup})}) \sum_{j=1}^{m} P_j \tag{3.14}$$

and

$$T'_{sup} = T_{sup} + T_{adj} \tag{3.15}$$

where

$$T_{adj} = T_{red} - max(T_{in}) \tag{3.16}$$

3.4 Illustrative Example

The following example illustrates that assigning VMs to minimum number of servers does not necessarily minimizes total power required. Location of each virtual machine (VM) should be selected based on its computational power and data center heat recirculation model to optimize the total power consumption in the computing nodes and the cooling infrastructure.

Suppose distribution vector $\vec{D}$ of data center shown in Fig. 2.4 is given as:

$$\vec{D} = \begin{bmatrix} 0.0723 & 0.0252 & 0.0071 & 0.0097 \\ 0.0128 & 0.0463 & 0.0009 & 0.0092 \\ 0.0157 & 0.0025 & 0.0272 & 0.0139 \\ 0.0035 & 0.0009 & 0.0076 & 0.0455 \end{bmatrix}$$

Actual distribution vector is obtained through computational fluid dynamic simulation on the given data center's configuration. Similarly, suppose $T_{sup} = 25^oC$, $P_{idle} = 150$ Watts, and $P_{busy} = 215$ Watts. The flow rate (f_i) of each server's fan is assumed to be 8.0 m^3/s. For the VMs assignment shown, Server 1 and 2 consume 215 Watts of power; whereas, server 3,4, and 5 are turned off. Using Equation 3.11. Inlet temperature (T_{in}) of these five servers are 46, 37, 29,

26, and 27 ^oC. The temperature (T_{sup}) of air supplied by the CRAC units must be 4^oC to ensure that $T_{in} \leq T_{red}$. Total power (P_{total}) consumed is given by Equation 3.14

$$P_{total} = (1 + \frac{1}{COP(4)})(215 + 215 + 0 + 0 + 0) = 1186.72\ Watts$$

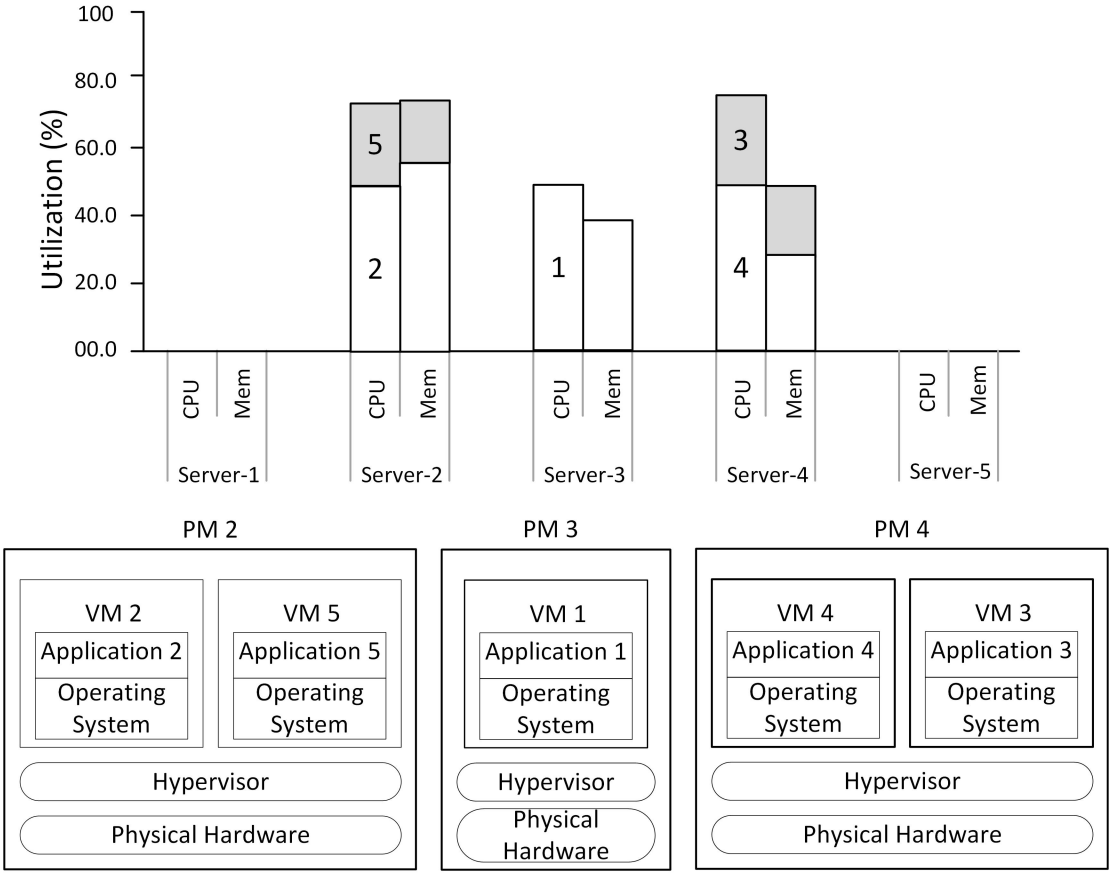


Figure 3.4: VMs allocated to avoid hot spots.

Now, consider VMs assignment shown in Figure 3.4. Server 2,3, and 4 are consuming 198.75, 182.50, and 198.75 Watts of power respectively; whereas, server 1 and 5 are turned off. Inlet temperature (T_{in}) of the five servers are 33, 36, 33, 35, 26 ^oC. The temperature (T_{sup}) of air supplied by the CRAC units must be

14^oC to ensure that temperature of the servers is within the safe limit.

Therefore

$$P_{total} = (1 + \frac{1}{COP(14)})(0 + 198.75 + 182.50 + 198.75 + 0) = 909.44\ Watts$$

This allocation of VMs is using more servers as compared to that of assignment shown in Fig. 2.4, but it is consuming less total power. Hence minimizing number of servers does not necessarily minimize total power. Computational power and cooling power must be optimized in an integrated fashion to minimize the total power consumption in data centers.

3.5 Problem Formulation

In this section, we formally define the virtual machine placement problem and discuss the cost function and constraints.

In this work, we consider two dimensions, CPU and memory to characterize a VM and PM. Suppose there are n VMs to be assigned. Every VM v_i, $i \in \{1, 2, 3, \ldots, n\}$ is defined as a 2-dimensional requirement vector, $v_i = \{v_i^c, v_i^m\}$ where each dimension represents a normalized value of one type of resource requested (CPU and memory). These VMs are to be allocated to n PMs with the assumption that every VM request can be satisfied by one sever. We assume a homogeneous data center, where all PMs have the same capacity. Let T_j^c and T_j^m be the threshold values of CPU and memory resources, associated with each PM

p_j, $j \in \{1, 2, 3, \ldots, m\}$ respectively. The assignment solution is represented by a $m \times n$ matrix X, where:

$$x_{i,j} = \begin{cases} 1: & \text{if } i^{th} \text{ VM is assigned to the } j^{th} \text{ PM} \\ 0: & \text{otherwise} \end{cases} \tag{3.17}$$

In addition, we define the following binary decision variable:

$$y_j = \begin{cases} 1: & \text{if } j^{th} \text{ PM is in use} \\ 0: & \text{otherwise} \end{cases} \tag{3.18}$$

The given problem can be formulated as:

$$\text{minimize } P_{total} = \left(1 + \tfrac{1}{COP(T'_{sup})}\right) \sum_{j=1}^{m} P_j$$

subject to

$$\sum_{i=1}^{n} v_i^c \times x_{ij} \leq T_j^c \times y_j \quad \forall j \in J \tag{3.19}$$

$$\sum_{i=1}^{n} v_i^m \times x_{ij} \leq T_j^m \times y_j \quad \forall j \in J \tag{3.20}$$

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in I \tag{3.21}$$

$$y_j, x_{ij} \in \{0, 1\} \quad \forall i \in I \quad and \quad \forall j \in J \tag{3.22}$$

$$T_{in} \leq T_{red} \tag{3.23}$$

Constraints (3.19) and (3.20) guarantee that the capacity threshold of each server is not exceeded. Moreover, constraint (3.21) ensures that a VM is placed in exactly one server and constraint (3.22) represents the domain of variables $x_{i,j}$ and y_j. Finally, constraint (3.23) makes sure that temperature of PMs is within safe limit.

CHAPTER 4

NON-DETERMINISTIC HEURISTICS

In this chapter, we describe three non-deterministic heuristics that are employed for optimal VM placement to minimize total power consumption in data centers.

4.1 Simulated Annealing

Simulated annealing (SA) is a well-established non-deterministic heuristic [42]. It has been adopted to solve various combinatorial optimization problems. One typical feature of SA is that, it accepts all the solutions with improved cost like a greedy algorithm but it also, to a limited extent, accepts changes which lead to inferior solutions. This feature gives SA the hill climbing capability that allows it to escape from the local optimal solutions initially and reach a more optimal solution at the end of the search. The odds of accepting deteriorated solutions are large in the beginning; but as the search progress, fewer bad solutions are

Algorithm Simulated_Annealing($X_0, T_0, \alpha, \beta, M, Max\_Time$);
(*X_0 is the initial assignment)
(*$Best\_X$ is the best assignment *)
(*T_0 is the initial temperature *)
(*α is the cooling rate *)
(*β a constant *)
(*Max_Time is the total allowed time for the annealing process *)
(*M represents the time until the next parameter update *)

Begin
1. $T = T_0$;
2. $Cur\_X = X_0$;
3. $Best\_X = Cur\_X$; /*$Best\_X$ is the best assignment */
4. $Current\_Cost = Cost(Cur\_X)$;
5. $Best\_Cost = Cost(Best\_X)$;
6. $Time = 0$;
7. **Repeat**
8. Call Metropolis;
9. $Time = Time + M$;
10. $T = \alpha T$;
11. $M = \beta M$
12. **Until** ($Time \geq Max\_Time$);
13. **Return**($Best\_X$)

End. (**of Simulated_Annealing**)

Figure 4.1: Procedure for simulated annealing algorithm.

accepted and finally solutions with improved cost are accepted. SA produces high quality solution regardless of the initial configuration. It is robust, effective, and easy to implement [43].

The simulated annealing algorithm is shown in Fig. 4.1. The *Simulated Annealing* routine takes as input the initial assignment matrix X_0, maximum time Max_Time, cooling rate α, and time until temperature update M. Temperature is initialized to T_0, and is reduced in a controlled fashion using cooling factor α. Standard Boltzmann-type SA uses logarithmic scheduling of temperature in which

Algorithm Metropolis(Cur_X, Cur_Cost, $Best\_X$, $Best\_Cost$, T, M);

Begin

1. **Repeat**
2. $New\_X = Neighbor(Cur\_X)$;
3. $New\_Cost = Cost(New\_X)$;
4. $\triangle Cost = New\_Cost - Cur\_Cost$;
5. **If** $(\triangle Cost < 0)$ **Then**
6. $Cur\_X = New\_X$;
7. **If** $New\_X < Best\_Cost$ **Then**
8. $Best\_X = New\_X$
9. **EndIf**
10. **Else**
11. **If** $(RANDOM(0,1) < e^{-\triangle Cost/T})$ **Then**
12. $Cur\_X = New\_X$=;
13. **EndIf**
14. **EndIf**
15. $M = M - 1$
16. **Until** $(M = 0)$

End. (******of Metropolis******)

Figure 4.2: The Metropolis procedure.

Temp (T) is selected to be not faster than

$$T_k = \frac{T_0}{\log k} \tag{4.1}$$

where k is the time index of annealing and T_0 is the initial temperature. SA technique is quite slow because of this logarithmic scheduling [44]. We are using a modified Simulated Annealing called Simulated Quenching (SQ) [45]. SQ solution methodology resembles the cooling process of molten metals through annealing. The algorithm and the analogy of the technique remains the same as that of SA

except for the annealing schedule. SQ uses an exponential schedule.

$$T_{k+1} = \alpha T_k \tag{4.2}$$

where α is the cooling factor.

The *Metropolis* routine is invoked after updating (lowering) temperature T. The amount of time for which search is performed at each temperature before cooling occurs is dictated by M. Parameter β is used to gradually increase M as temperature is decreased. The variable *Time* keeps track of the time being spent in each call to the *Metropolis* routine. The annealing procedure halts when *Time* exceeds the allowed time.

The *Metropolis* routine shown in Fig. 4.2 is an essential part of the algorithm [46]. It takes as input the current assignment Cur_X, and current temperature T. It improves the Cur_X through local search by simulating annealing at given temperature T. It also receives M, which is amount of time spent by *Metropolis* at temperature T. It uses the procedure *Neighbor* to perturb the any assignment X to generate a local neighbor New_X. Neighbor function perturb the current assignment by randomly assigning a VM to a server. It is ensured that the new assignment is feasible. The *Cost* function is used to compute the cost of any given assignment X. If cost of the new assignment New_X is better than the cost of the current assignment Cur_X, then the new assignment is accepted and Cur_X is replaced by New_X. If the cost of the new assignment is better than the best assignment ($Best\_X$) seen thus far, then we also replace

$Best\_X$ by New_X. If the new assignment has a higher cost in comparison to the original assignment Cur_X, *Metropolis* will accept the new assignment on a *stochastic* basis. The probability that an inferior assignment is accepted by the *Metropolis* is given by $P(RANDOM(0,1) < e^{-\triangle Cost/T})$. $RANDOM(0,1)$ is used to generate random numbers in the range 0 to 1. If $e^{-\triangle Cost/T}$ is larger than this random number than inferior assignments are accepted, where $\triangle Cost$ is the difference in costs. The random number generation is assumed to follow a *uniform distribution*. At very high temperatures, (when $T \rightarrow \infty$), $e^{-\triangle Cost/T} \simeq 1$, and hence the above probability approaches 1 and every bad assignment is accepted. On the contrary, when $T \rightarrow 0$, the probability $e^{-\triangle Cost/T}$ falls to 0 and only good assignments are accepted.

4.2 Tabu Search

Tabu Search (TS) is a general iterative heuristic introduced by Fred Glover [47] [48] for solving combinatorial optimization problems. Tabu Search is a generalization of local search. This scheme works by moving from one solution to another in a hill climbing fashion. Unlike local search which stops when no improved new solution is found in the current neighborhood, Tabu Search continues the search from the best solution in the neighborhood even if it is worse than the current solution. One of its features is its systematic use of *adaptive* (flexible) memory. Tabu Search differs from genetic algorithm which are "memoryless", and also from branch-and-bound, A* search, etc., which are rigid memory approaches. An

algorithmic description of a simple implementation of the Tabu Search is given in Fig. 4.3.

Algorithm Tabu Search

Ω : Set of feasible assignments.
X : Current assignment.
$\aleph(X)$: Neighborhood of $X \in \Omega$.
X^* : Best admissible assignment.
$\mathbf{V}^*$: Sample of neighborhood assignments.
$Cost$: Objective function.
AL : Aspiration Level.
TL : Tabu List.

```
Begin
1.  Start with an initial feasible assignment X ∈ Ω.
2.  Initialize Tabu List TL.
3.  For fixed number of iterations
4.      Generate neighbor assignments V* ⊂ ℵ(X).
5.      Find best X* ∈ V*.
6.      IF move X to X* is not in TL  Then
7.          Accept move and update best assignment.
8.          Update TL and AL.
9.          Increment iteration counter.
10.     Else
11.         IF Cost(X*) < AL Then
12.             Accept move and update best assignment.
13.             Update TL and AL.
14.             Increment iteration counter.
15.         End If
16.     End If
17. End For
    End.
```

Figure 4.3: Algorithmic description of short-term Tabu Search (TS).

The procedure starts from an initial feasible assignment X (current assignment) in the search space Ω. A neighborhood $\aleph(X)$ is defined for each X. A sample of neighbor assignment $\mathbf{V}^* \subset \aleph(X)$ is generated called *trail* assignments ($n = |\mathbf{V}^*| \ll |\aleph(X)|$), and comprises what is known as candidate list. From this

generated set of assignments, the best assignment, say $X^* \in \mathbf{V}^*$, is chosen for consideration as the next assignment. The move to X^* is considered even if X^* is worse than X, that is, $Cost(X^*) > Cost(X)$.

Selecting the best move in $\mathbf{V}^*$ is based on the assumption that good moves are more likely to reach optimal or near-optimal assignment. As mentioned above, the best candidate assignment $X^* \in \mathbf{V}^*$ *may* or *may not* improve the current assignment, but is still considered. It is this feature that enables *escaping* from local optima. However, even with this strategy, it is possible to trap in a local optimum. Search will ascend (in case of a minimization problem) since moves with $Cost(X^*) > Cost(X)$ are accepted, and then in a later iteration return back to the same local optimum. That is, there is a possibility of *cycling* by returning back to previously visited assignments. This may cause the search to go through the same subset of assignments for ever.

A Tabu List is maintained to prevent returning to previously visited assignments. When a move is accepted, its attributes are introduced into the Tabu List. The purpose is to prevent the reversal of moves for the next $k = |\mathbf{T}|$ iterations because they *might* lead back to a previously visited assignment.

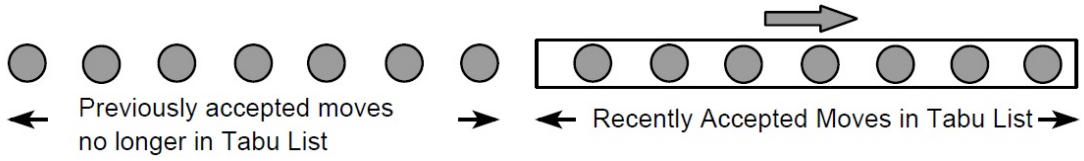


Figure 4.4: Tabu List.

In certain situations, it is necessary to overrule the tabu status. This is done with the help of the notion of *aspiration criterion*. Aspiration criterion overrides

the tabu status of moves whenever appropriate. We are using *best assignment* criterion that overrides the tabu status if the move produces an assignment better than the best obtained thus far

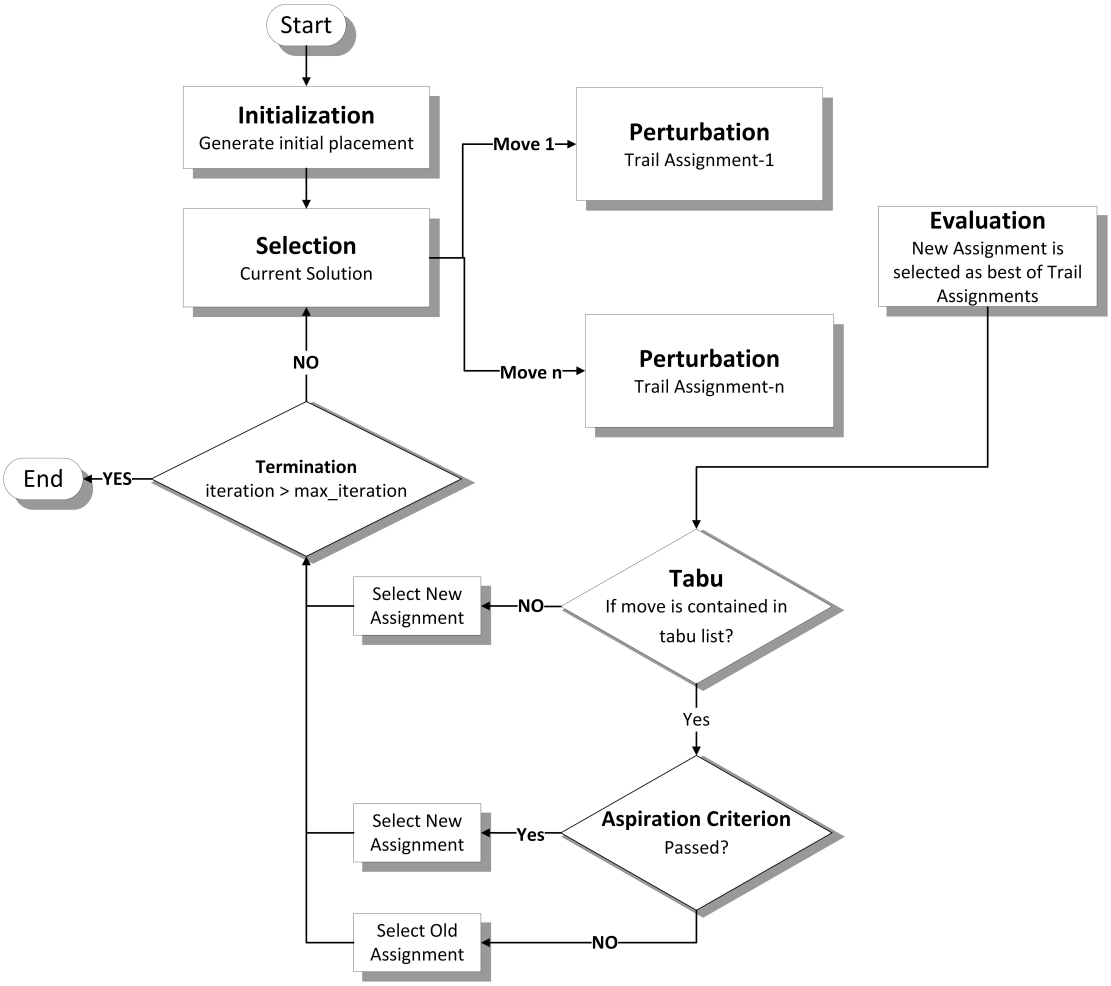


Figure 4.5: Flow chart of TS.

Any constructive method can be used to generate a feasible initial assignment. In our case, the initial assignment matrix X generated by randomly assigning VMs to the servers using admission control. As search proceeds, neighbor assignments are generated by moving some VM to a server. The selection of VM and server is done randomly. Maintaining a Tabu List is an important step in TS. We are

using only one Tabu List. Each entry in the Tabu List contains the following information:

- VM that was selected for the move,
- Server *from* where VM is being moved,
- Server *to* where VM is being moved.

After a successful move, inverse of the move is stored in Tabu List. Every new move is checked against the moves stored in the Tabu List to make sure no reverse moves are being made, and heuristics can escape local optimal. However, Tabu List moves are ignored if aspiration criterion is met.

4.3 Simulated Evolution

Simulated Evolution (SimE) was proposed by Kling and Banerjee in 1987 [49]. The algorithm combines constructive perturbation and iterative improvement and save itself from getting stuck to the local minima by following a stochastic approach. In SimE, the search space is traversed by making intelligent moves, unlike *TS* and *SA* where random moves are made. The core of the algorithm is the goodness estimator. SimE assigns each moveable element a goodness value. The goodness value indicates how well a certain movable element is currently assigned. The more the goodness value, the lesser is the probability of the element being selected for re-allocation.

The flow of our proposed (SimE) algorithm is shown in Fig. 4.6. SimE starts with an initial solution Φ of a set V containing n movable elements (VMs). It, then

follows an evolution-based approach to find better solutions from one iteration to the next by perturbing some ill-assigned elements (VMs) while retaining the near-optimal ones. The algorithm consists of three sequential steps, **evaluation**, **selection** and **allocation** that are executed in each iteration. The process of iterative improvements continues until the solution average goodness value reaches at its maximum, or no considerable improvement in solution quality is observed after a given number of iterations [43].

4.3.1 Goodness Evaluation

This step involves the evaluation of goodness (fitness) g_i of each VM v_i assigned to PM p_k in current solution Φ'. Effective goodness measures can be thought of based on the domain knowledge of the optimization problem [50]. This goodness measure is expressed as a single number in the range of zero to one. For our VM assignment problem, we used a joint goodness function that is based on our objective; i.e., to reduce computational and cooling power.

Computational Power is direct proportional to the number of active severs as given by Equation 3.3. We can save computational power by assigning the given VMs to fewer number of servers. We define a goodness value gs of i^{th} VM assigned to k^{th} server as:

$$gs_i = \frac{v_i^c + v_i^m}{p_k^c + p_k^m} \tag{4.3}$$

where v_i^c and v_i^m are CPU and memory requirements of VM v_i, and p_k^c and p_k^m are the available CPU and memory resources of partially used PM p_k after removing

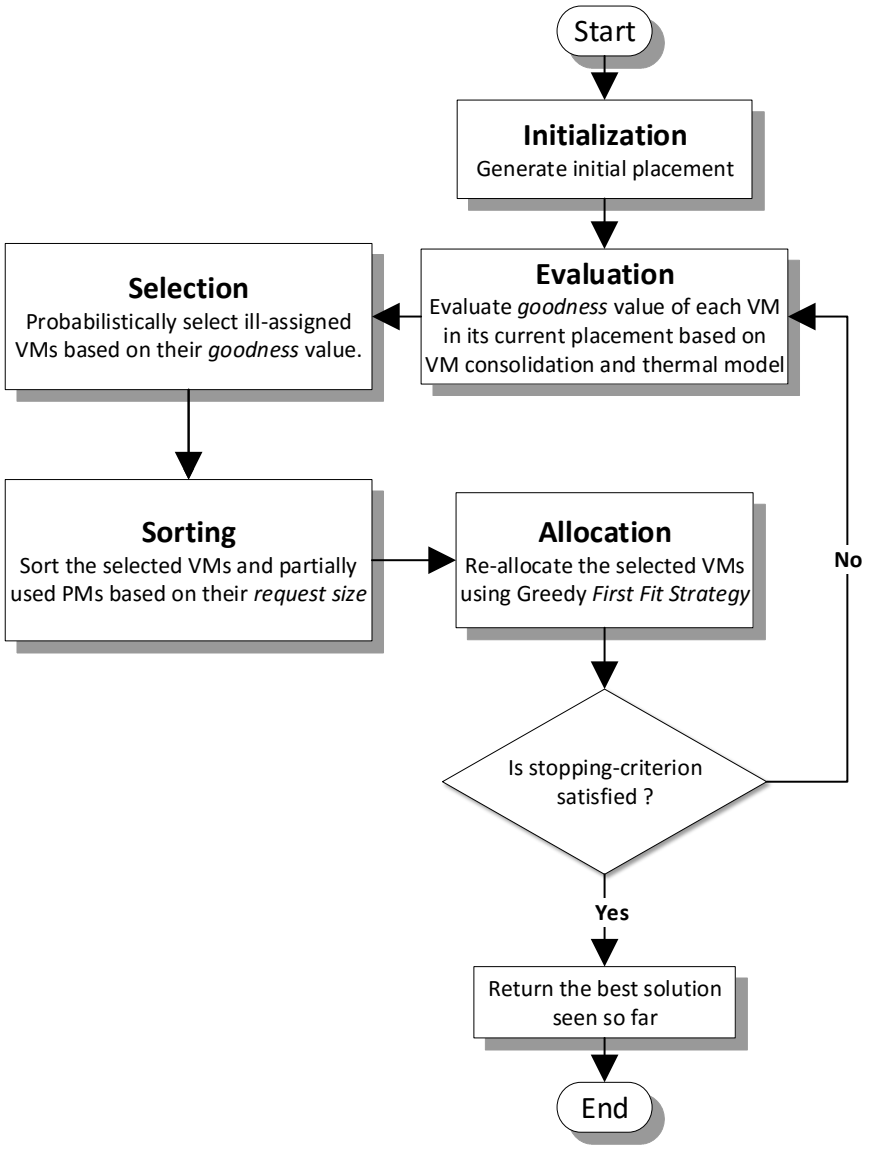


Figure 4.6: Flowchart of SimE.

VM v_i from PM p_k in the current solution Φ'. Equation (4.3) assumes a minimization of resource wastage in PM p_k (maximization of goodness). The goodness of a VM v_i will be 1 if it is assigned to such a partially used PM p_k that $v_i^c = p_k^c$ and $v_i^m = p_k^m$. It means that the current assignment of VM v_i exactly packs the PM p_k and hence optimally utilizes the PM p_k [29]. The objective of this goodness measure is to minimize the number of active servers.

Our other objective is to minimize cooling power. Hot spots are created in data centers due to heat recirculation. Some of these recirculation effects can lead to situations where the observed consequence of the inefficiency is spatially uncorrelated with its cause; in other words, the heat vented by one machine may travel several meters before arriving at the inlet of another server. From Equation 3.11, we can define increment in the inlet temperature ($\delta\overrightarrow{T_{in}}$) of the severs due to heat recirculation as:

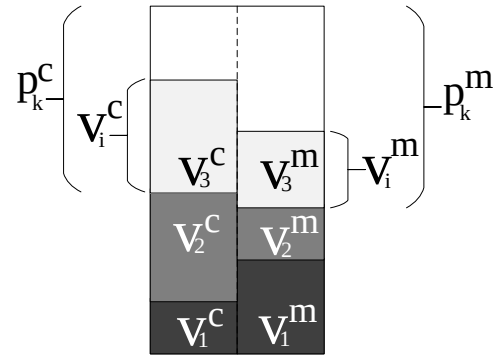
$$\delta\overrightarrow{T_{in}} = \mathbf{D} \times \overrightarrow{P} \tag{4.4}$$

where $\delta\overrightarrow{T_{in}} = \{\delta T_{in}^1, \delta T_{in}^2, \delta T_{in}^3, \ldots, \delta T_{in}^m\}$ and $\mathbf{D}$ is given by Equation. 3.12. Similarly, increment in the inlet temperature of i^{th} server due to heat recirculation of j^{th} server is given as:

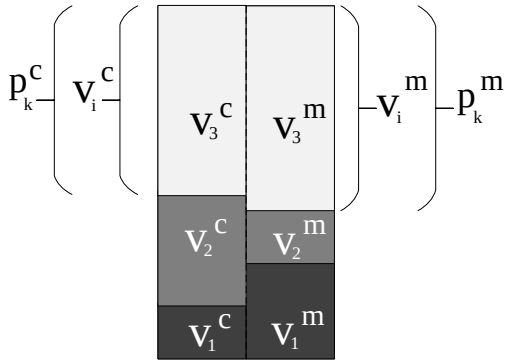
$$\delta\overrightarrow{T_{in}^i}(j) = \overrightarrow{d_{i,j}} \times \overrightarrow{p_j} \tag{4.5}$$

We define Recirculation Effect (RE) of j^{th} server as:

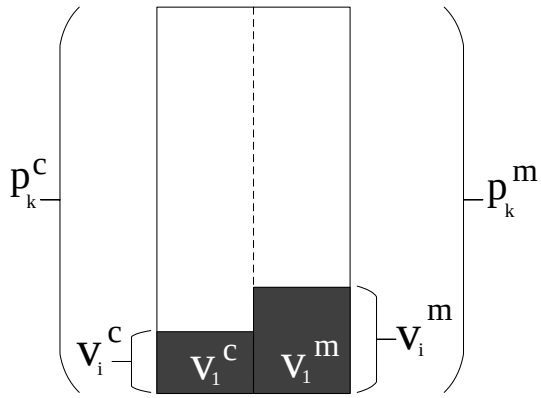
$$RE_j = \sum_{i=1}^{m} \delta\overrightarrow{T_{in}^i}(j) = \sum_{i=1}^{m} \overrightarrow{d_{i,j}} \tag{4.6}$$



(a)



(b)



(c)

Figure 4.7: (a) Goodness measure for minimizing number of servers $gs_i = \frac{v_i^c + v_i^m}{p_k^c + p_k^m}$ (b) The 3^{rd} VM has a goodness measure of 1 and it should not be selected for re-allocation (c) The 3^{rd} VM has a goodness value of 0, it is ill assigned and it should be reallocated.

RE indicates the contribution of a server to heat recirculation and creation of hot spots. In order to avoid hot spots and to minimize cooling power (P_{AC}), servers with higher RE value must be avoided. RE value can be used to define the goodness value of VMs assigned to a server. If VMs are assigned to the servers with higher RE value, they should have lower goodness value and should be considered for re-allocation. Since the goodness measure must be a single number expressible in the range [0,1]. We translate RE to a goodness measure gt as:

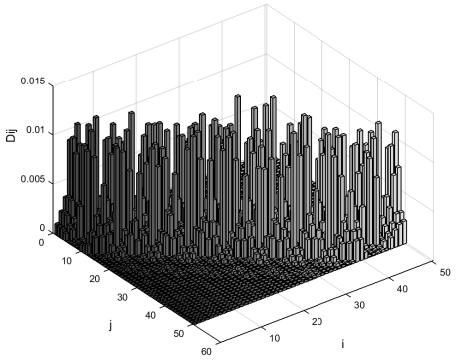
$$gt_i = 1 - \frac{RE_j - min(RE)}{max(RE)}, \quad gt_i \in [0, 1] \tag{4.7}$$

where gt_i is the goodness of i^{th} VM assigned to j^{th} server. The goodness value gt focuses on inefficiencies; i.e., it will lower the total amount of heat that recirculates within the data center.

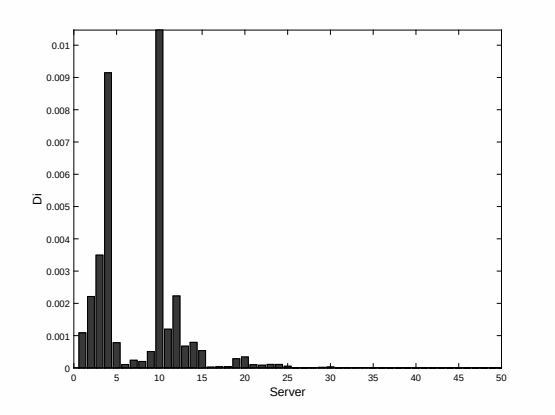
Overall goodness value used by our algorithm is be defined as:

$$g_i = \alpha \times gs_i + \beta \times gt_i, \quad g_i \in [0, 1] \tag{4.8}$$

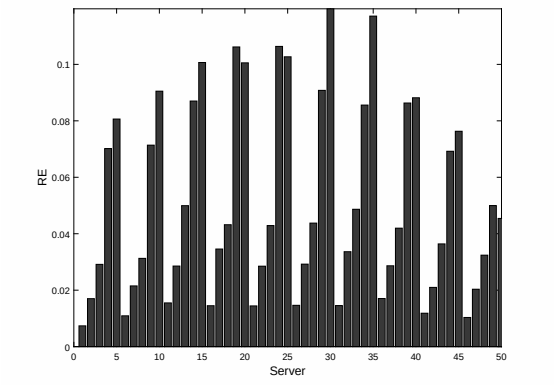
where α and β are constant ranging from 0 to 1 and $\alpha + \beta = 1$. The goodness function given in Equation 4.8 strongly reflects the target objectives of the given problem. The quality of a solution can also be estimated by summing up the goodness of all the VMs.



(a)



(b)



(c)

Figure 4.8: (a) A sample distribution matrix for fifty servers (b) A distribution vector for 1^{st} server (c) Recirculation effect (RE) values of fifty servers.

ALGORITHM
Simulated_Evolution$(V,\ Stopping-criteria)$;
/* $\mathbf{\Phi_i}$: Initial Solution; */
/* $\mathbf{\Phi_p}$: Partial Solution; */
/* $\mathbf{\Phi'}$: New Solution; */
/* $\mathbf{V}$: Set of all VMs, where $|V| = n$; */
/* $\mathbf{V_s}$: Selected VMs for re-allocation; */
/* $\mathbf{P_a}$: Active PMs in Φ_p; */
/* $\mathbf{B}$: Selection bias; */
/* **maxSelection**: Upper limit of the selection set size; */
$INITIALIZATION$;
    $\Phi_i = initial\_placement(V)$;
    $\Phi' = \Phi_i$;
Repeat
$EVALUATION$:
    ForEach $v_i \in V$ **Do**
        $g_i = Evaluate(v_i)$; /* *Evaluate using goodness estimator* (*Equation* 4.8) */
    EndForEach;
$SELECTION$:
    $\Phi_p = \Phi'$;
    $counter = \mathbf{0}$;
    ForEach $v_i \in V$ **Do**
        If $(Random \leq (1 - g_i + B)) \wedge (counter \leq maxSelection)$ **Then**
            $V_s = V_s \cup \{v_i\}$;
            $\Phi_p = \Phi_p - \{v_i\}$;
            $counter = counter + 1$;
        EndIf;
    EndForEach;

$ALLOCATION$:
Sort the VMs in set V_s *based on their resource demand* ;
    ForEach $v_i \in V_s$ **Do**
        Allocate$(\ v_i\ , \Phi_p)$; /* *Allocate* v_i *in* Φ_p, *using Greedy First Fit Strategy* */
    EndForEach;
        $\Phi' = \Phi_p$;
Until *Stopping-criterion is satisfied*;
Return $(BestSolution)$;
End *Simulated_Evolution.*

Figure 4.9: Simulated Evolution Algorithm for VM assignment.

4.3.2 Selection

In this step, elements are selected for relocation probabilistically. Elements with lesser *goodness* values have more probabilities of getting selected. This step divides Φ' into two disjoint sets; a set V_s of selected elements and a partial solution Φ_p containing rest of the elements of the solution Φ'. Every element of the solution is considered separately from all other elements. The decision of selecting an element v_i to the set V_s depends on its goodness g_i. The selection operator has a non-deterministic nature, i.e., an individual with a high goodness (close to one) still has a non-zero probability of being assigned to the selection set V_s. It is this element of non-determinism that makes SimE capable of escaping local minima. Each time a VM v_i is considered for selection a random number is generated. The inequality $Random \leq (1 - \ g_i \ + B)$ is used for this purpose (see Fig. 4.9). Error in goodness estimation is compensated by using a selection *bias* (B). The objective of this bias value is to deflate or inflate the goodness of elements. A high positive value of bias increases the probability of selection while the negative value has the opposite effect. Large selection sets may lead to better solution, but will require higher run time. On the other hand, small selection sets will speed-up the algorithm, but with the risk of an early convergence to a sub-optimal solution (local minima). Values of B are recommended to be in the range [-0.2, 0.2]. In many cases a value of $B = 0$ would be a reasonable choice as in our case [43]. In addition to bias value, $maxSelection$ also provides control over the selection process and restricts the maximum size of the selection set. In this work, a value

of 40% of the total number of items was adopted. This keeps the time requirement of the SimE algorithm under control, especially during the allocation step, which is the most time consuming step of the algorithm.

4.3.3 Allocation

The allocation step has great influence on solution quality. Allocation takes the elements of set V_s and the partial solution Φ_p and generates a complete new solution Φ' with the elements of set V_s mutated according to allocation strategy. The goal of *Allocation* strategy is to favor improvements over the previous iteration, without being too greedy [43]. Superior alterations gradually improve the individual goodness values as the goodness of each individual element also reflects the target objective. Hence, *Allocation* helps the search to progressively converge towards a target configuration where every individual is optimally located.

The design of allocation strategy is problem specific. Just like in the design of goodness function, the choice of allocation strategy also requires ingenuity on the part of the designer. In this work we adopted a variant of FFD heuristic as our allocation strategy. The VMs selected during the *selection* step are sorted in decreasing order of their request sizes (Rv_i) computed using Equation (4.9).

$$Rv_i = (v_i^c)^2 \ + \ (v_i^m)^2 \tag{4.9}$$

Subsequently, First Fit algorithm is applied to generate the new solution Φ'. This algorithm assigns the selected VMs to the servers with low RE value.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, we provide performance evaluation of our proposed approach with those in literature. We compare it with the improved version of classic First Fit Decreasing (FFD_{imp}) and Least Loaded (LL_{imp}) algorithms proposed by Arijto *et al.* [33]. These algorithms try to minimize power consumption in data centers by packing VMs in minimum number of servers.

5.1 Baseline Algorithms

First Fit Decreasing (FFD) and its variants are common deterministic methods to find an approximate assignment to the vector bin-packing problem [51]. FFD algorithm first sorts the VMs in decreasing order of their sizes and then places them in PMs according to First Fit (FF) strategy. In the context of data center power optimization, Lei Shi *et al.* [51] presented and evaluated the performance of six different FFD-based VBP algorithms. Panigrahy *et al.* [52] systematically studied the family of FFD heuristics and their limitations and suggested a new

geometric based heuristic approach for VBP.

Ajiro *et al.* [33] suggested improvements to the classical FFD and least loaded (LL) algorithms. Improved FFD (FFDimp) is different from conventional FFD in the sense that it seeks near-optimal assignment in multiple passes. In the first pass VMs are sorted in decreasing order of their highest resource demand, i.e., sum of all CPU demands or sum of all memory demands, whichever is larger. Then an attempt is made to pack all the VMs in number of PMs equal to theoretical lower bound. If any VM cannot be packed, then it is moved to a priority queue and placement process is aborted. In the next pass, VMs in the priority queue are placed first, followed by the remaining VMs in the sorted list. The above steps are repeated MAXR times. If all VMs are not packed in MAXR iterations then the number of destination PMs is incremented by one and the above process is repeated until an assignment is found, i.e., all VMs are packed. LLimp is implemented in multi-passes in a similar way employing the LL heuristic. These improved versions provide better quality assignments than that of their single-pass implementations. This improvement, however, comes at the expense of increased run time [33].

5.2 Experimental Setup

Programs for the proposed $SimE$, SA, TS, FFD_{imp}, and LL_{imp} heuristics are coded in MATLAB. The simulations are run on a computer equipped with Intel $core^{TM}$ i3 with 2.4GHz CPU and 6GB RAM.

For simplicity, we assume a homogeneous hardware environment. All servers have the same power consumption and computing capability. We are using $\bar{p}_{busy} = 215$ Watts and $\bar{p}_{idle} = 162$ Watts as reported by Gao *et al.* [28] through experiments conducted on a Dell server. We are using 150 servers and 1 CRAC unit. The flow rate (f_i) of each server's fan is set to 8.0 m^3/s.

SA has four parameters which need to be tuned carefully as explained in Section 4.1. After trail runs, appropriate values of these parameters are found to be $T_0 = 800$, $\alpha = .98$, $\beta = 1.1$, and $M = 9$. Similarly, parameters of TS are set as $TabuListSize = 10$ and $MaxTrailAssignments = 15$. SimE is set to stop after 30 iterations if there is no significant improvement in cost and $maxSelection$ is set to 40% of total VMs. For our goodness estimator, α and β both are set to 0.5.

5.2.1 Work Load

The problem instances were a set of two resource demand vectors representing the CPU and memory utilization of VMs. Servers were assumed to be identical, that is, all PMs have the same resource capacity fixed at 90% although the proposed approach is equally applicable for the heterogeneous case. Due to non-deterministic behavior, average of results obtained from 20 independent runs are reported. Suitably parametrized random ensembles of instances of problems are used. In this context, it was observed that in some regions of the ensemble space instances are typically easy to solve, while in other regions instances are found to be typically hard. To make synthetic instances more representative and

for $i = 1$ **to** n
  $v_i^c \leftarrow$ **rand**$(2\bar{v^c})$
  $v_i^m \leftarrow$ **rand**$(\bar{v^m})$
  $r =$ **rand**$(0, 1)$
  if $(r < P$ **and** $v_i^c \geq \bar{v^c})$ **or** $(r \geq P$ **and** $v_i^c < \bar{v^c})$
    $v_i^m \leftarrow v_i^m + \bar{v^m}$
  end if
end for

Figure 5.1: Pseudocode to generate different problem instances with certain correlations.

cover a wide range of possible workloads, we generated problem instances with two different average resource values and several correlations of CPU and memory utilization, employing the method proposed by Ajiro *et al.* [33]. The pseudo code for this is given in Fig. 5.1. rand(0, 1) is a function that returns uniformly distributed random real numbers in the range [0, 1); v^c denotes the average CPU utilization while v^m represents the average memory utilization. The probability P is used to decide whether both the utilization of CPU and memory would be equal to or higher than the average values, or both utilizations would be lesser than the average values. By varying this probability P, we can control the correlations of CPU and memory utilization to some extent.

In our experiment, we used two kinds of average values and five different probabilities. We set both v^c and v^m to 25%, and then to 45%. The distribution of CPU and memory utilization were in the range of [0, 50%) when $v^c = v^m = 25\%$, and [0, 100%) when $v^c = v^m = 45\%$. For v^c and $v^m = 25\%$ we set P equal to 0.00, 0.25, 0.50, 0.75, and 1.0, and for this the average correlation coefficients obtained are -0.7485, -0.3813, 0.0081, 0.3736, and 0.7493 for each set of instances. These coefficients correspond to strong-negative, weak-negative, no, weak-positive, and

strong-positive correlations. The same values of P were used for v^c and $v^m = 45\%$ and then the correlation coefficients were -0.7508, -0.3703, 0.0019, 0.3857, and 0.7476. Threshold values of both utilizations were kept at $T_k^c = T_k^m = 90\%, k \in \{1, 2, 3, \ldots, m\}$ throughout these experiments.

5.3 Results and Discussion

Power reduction of $SimE$, SA, and TS over FF_{imp}, and LL_{imp} at 50% and 70% loading for 25% and 45% reference value is shown in Table 5.1. The measures for comparison are average power consumption (P_{total}), maximum inlet temperature (T_{in}), number of active servers (m), and CPU time (T).

From Table 5.1 the following observations can be made:

- For all algorithms, total power increases by increasing average resource value from 25% to 45%.
- Similarly, total power consumption increases by increasing the loading from 50% to 70%. This is because it is difficult to avoid hot spots with increased loading.
- The timing performance of SA, FFD_{imp} and LL_{imp} strongly depends on the correlation between CPU and memory utilization. On the other hand, execution time of $SimE$ and TS varies slightly across different correlation.
- TS gives better results as compared to SA but requires more CPU time.
- In each case $SimE$ outperforms other algorithms while requiring less execution time.

Breakdowns of total power P_{total} into computational power P_{CN} and cooling power P_{AC} at various correlations is shown in Fig. 5.2. It can be observed that, for all correlations, total power consumption of $SimE$ is the lowest. Similarly, CPU time at various correlation is illustrated in Fig. 5.3.

Table 5.1: Comparison of the SimE, SA, and TS with other techniques.

			50% Loading				70% Loading			
Reference value	**Corr.**	**Algorithm**	$\vec{P_{total}}$	$max(\vec{T_{in}})$	**s**	**T(sec)**	$\vec{P_{total}}$	$max(\vec{T_{in}})$	**s**	**T(sec)**
$\overline{v^c} = \overline{v^m} = 25\%$	strong +ve	FFD_{imp}	43.09	43.68	89.00	28.80	59.20	43.69	122.00	33.52
		LL_{imp}	41.11	43.26	87.00	10.90	57.94	43.66	119.00	8.85
		SA	33.53	37.28	102.75	215.31	55.51	41.28	137.10	80.88
		TS	30.65	36.13	97.40	181.80	54.25	41.97	125.70	237.60
		SimE	25.40	31.35	90.45	4.55	49.42	40.49	123.65	8.03
	weak +ve	FFD_{imp}	43.66	43.69	90.00	45.21	60.96	43.68	126.00	51.41
		LL_{imp}	41.07	43.45	85.00	6.91	58.53	43.75	119.00	6.97
		SA	34.09	37.39	103.80	215.61	56.36	41.37	138.10	80.87
		TS	30.80	36.09	97.80	178.92	53.52	41.50	127.50	381.23
		SimE	24.24	30.57	87.85	3.93	44.69	39.44	118.70	7.31
	zero	FFD_{imp}	47.60	43.69	100.00	47.71	64.88	43.79	135.00	84.26
		LL_{imp}	42.71	43.20	91.00	10.52	59.68	43.64	123.00	14.54
		SA	36.41	38.08	108.00	125.79	57.71	41.74	138.10	80.64
		TS	32.46	36.43	102.20	183.50	55.43	41.90	129.00	228.70
		SimE	23.98	30.51	87.15	3.69	43.83	39.08	117.95	6.94
	weak -ve	FFD_{imp}	48.47	43.68	102.00	33.14	67.70	43.75	143.00	121.74
		LL_{imp}	42.93	43.23	91.00	6.20	60.01	43.56	125.00	22.06
		SA	36.69	38.08	108.55	78.06	56.28	41.30	138.70	83.62
		TS	33.14	36.55	104.00	186.72	50.42	40.05	132.50	257.67
		SimE	23.14	30.29	84.05	3.06	39.44	37.17	116.05	6.53
	strong -ve	FFD_{imp}	51.43	43.68	109.00	43.26	71.32	43.82	150.00	134.78
		LL_{imp}	44.02	43.26	93.00	4.94	61.62	43.71	126.00	9.89
		SA	36.44	37.95	107.80	81.32	58.67	41.67	140.40	80.66
		TS	33.14	36.55	104.00	186.72	52.02	40.18	135.00	259.56
		SimE	23.36	30.28	85.55	3.02	39.82	37.39	116.70	6.57
$\overline{v^c} = \overline{v^m} = 45\%$	strong +ve	FFD_{imp}	41.30	43.56	86.00	14.33	63.60	43.71	131.00	20.59
		LL_{imp}	39.84	43.17	85.00	6.86	62.42	43.61	129.00	7.37
		SA	27.96	34.43	94.40	84.61	58.60	41.70	139.70	74.00
		TS	27.18	33.65	93.90	168.99	51.74	40.19	133.80	197.60
		SimE	32.54	35.47	105.9	2.28	67.27	42.82	150.0	2.63
	weak +ve	FFD_{imp}	46.56	43.57	98.00	25.74	67.94	43.33	145.00	35.36
		LL_{imp}	45.25	43.14	98.00	14.68	67.42	43.62	140.00	9.79
		SA	32.61	35.80	105.70	84.22	65.49	42.58	147.40	75.16
		TS	31.61	35.02	105.30	178.48	62.14	42.28	141.50	197.34
		SimE	28.63	34.06	97.40	1.92	64.80	42.65	144.8	4.93
	zero	FFD_{imp}	47.92	43.57	100.00	19.24	69.75	43.63	145.00	23.11
		LL_{imp}	48.19	43.42	102.00	11.80	69.68	43.53	146.00	10.75
		SA	34.77	36.47	109.25	81.48	67.27	42.76	149.10	75.40
		TS	33.21	35.57	107.60	174.95	65.63	42.58	146.70	217.10
		SimE	30.85	35.10	100.25	2.48	66.91	42.89	146.45	4.60
	weak -ve	FFD_{imp}	47.27	43.27	103.00	42.04	69.65	43.34	150.00	60.10
		LL_{imp}	46.77	43.47	100.00	17.05	70.07	43.63	147.00	17.46
		SA	33.17	36.05	107.35	81.14	65.58	42.54	148.30	77.40
		TS	32.29	35.44	106.80	168.61	64.74	42.45	147.10	194.34
		SimE	29.05	34.40	97.25	2.46	61.37	42.22	139.85	5.67
	strong -ve	FFD_{imp}	49.75	43.26	108.00	45.59	69.48	43.25	150.00	60.33
		LL_{imp}	50.15	43.60	106.00	19.22	71.26	43.62	150.00	19.15
		SA	35.72	36.59	112.75	88.22	67.51	42.87	149.40	87.72
		TS	35.75	36.62	112.70	145.98	68.20	42.98	149.90	228.75
		SimE	22.65	30.24	83.10	2.19	50.24	40.38	127.15	4.69

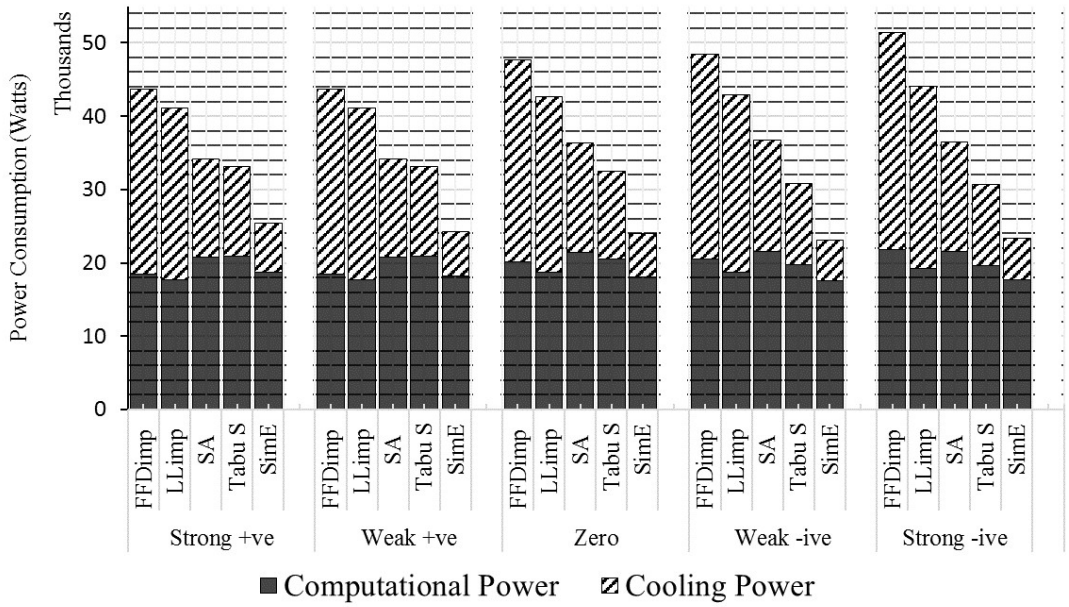
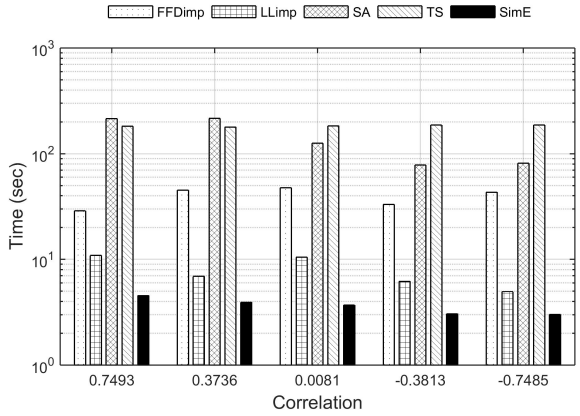
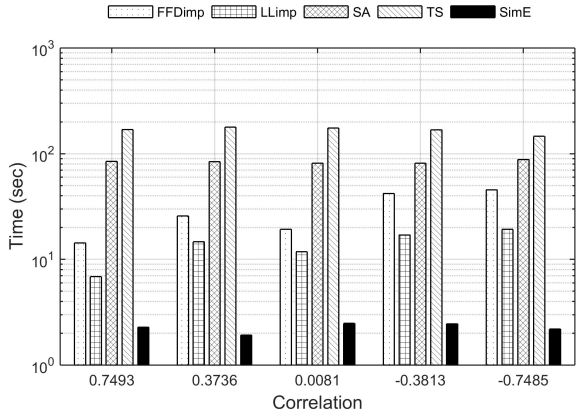


Figure 5.2: Power breakdown for different algorithms at various correlations for cases of $\overline{v^c} = \overline{v^m} = 25\%$.

SA and TS perform better than FFD_{imp} and LL_{imp}. FFD_{imp} and LL_{imp} attempt to assign VMs to a minimum number of servers. They do not consider heat recirculation in the data center; therefore, resultant assignment reduces computational power but creates hot spots due to higher utilization per server. On the other hand, SA and TS optimize total power by considering the trade-off between computational and cooling power, but they require significant CPU time. $SimE$ is performing better than TA and SA because $SimE$ uses its goodness function to direct the search; whereas TS and SA use hill climbing, memory and other features to find the optimal assignment but they lack intelligent moves. The precise selection of ill-assigned VMs and proper re-allocation plays a key role in improving the solution quality and reducing run time. Although SA, TA, and SimE all are iterative non-deterministic heuristics, SimE is more intelligent and thus requires fewer iterations to converge towards a desirable solution



(a) $\overline{v^c} = \overline{v^m} = 25\%$.



(b) $\overline{v^c} = \overline{v^m} = 45\%$.

Figure 5.3: Run time of FFD_{imp}, LL_{imp}, SA, TS and SimE with 50% loading for cases of (a) $\overline{v^c} = \overline{v^m} = 25\%$ and (b) $\overline{v^c} = \overline{v^m} = 45\%$.

Fig. 5.4 shows total power consumption of various schemes at different loadings of the data center. It is reported for 25% reference value and strong negative correlation. It is evident that $SimE$ is performing better than other schemes for most of the loading span.

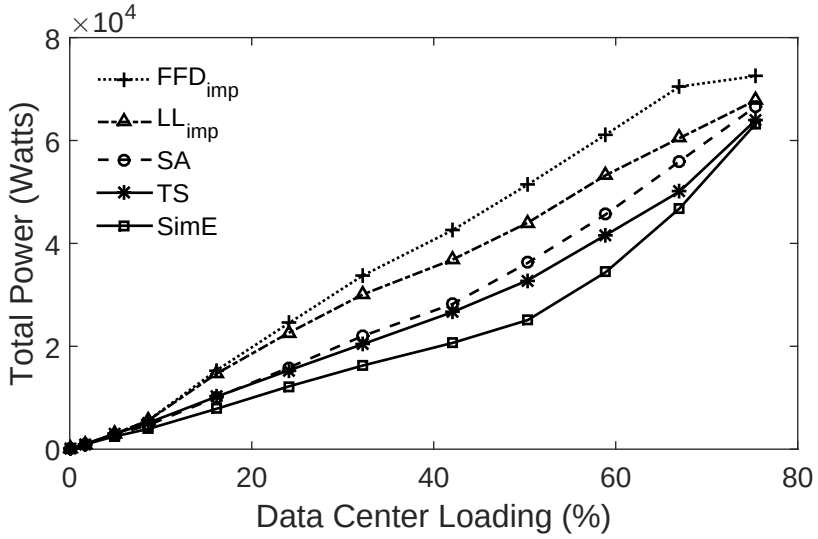
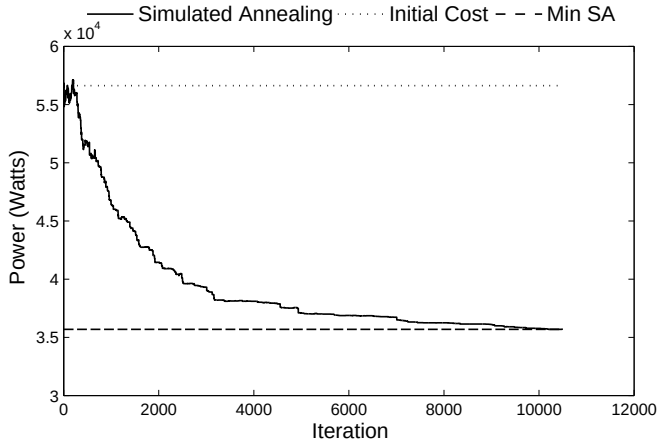


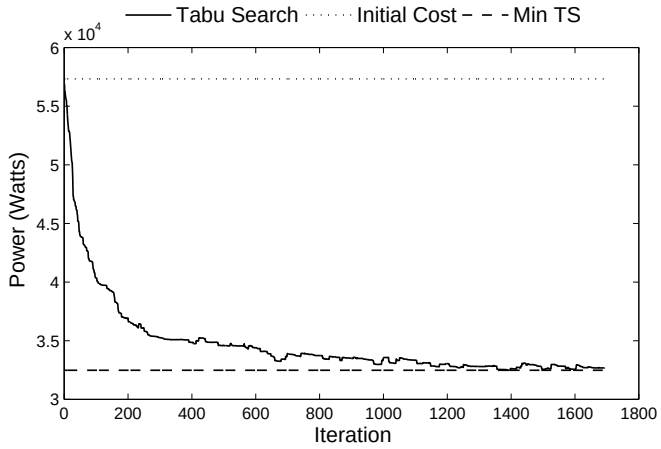
Figure 5.4: Comparison of Algorithms for various loadings for cases of $\overline{v^c} = \overline{v^m} = 25\%$.

Change in total power, max inlet temperature, and number of active servers of $SimE$, SA and TS with iterations are illustrated in Fig. 5.5, 5.6, and 5.7 respectively. Without the loss of generality, the case reported has the following values: 50% loading, $R_c = R_m = 25\%$, and strong negative correlation. However, similar results are obtained for other cases. From graphs, it is evident that $SimE$ escapes local minima multiple times by making intelligent moves and attempts to reduce total power.

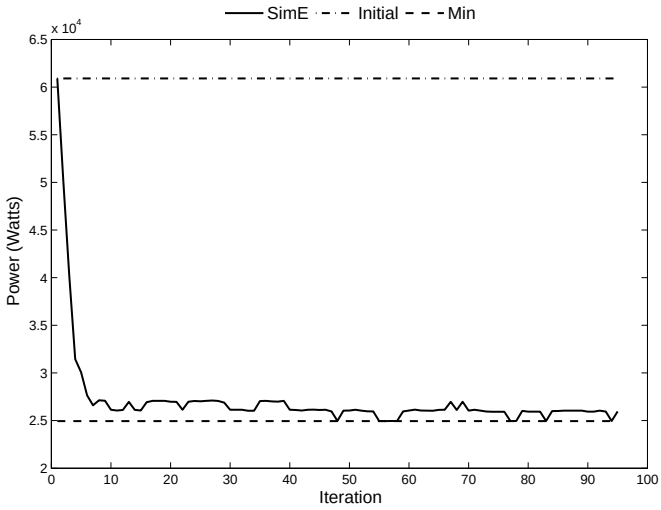
Finally, as it can be seen in Fig. 5.8, the overall average goodness of all movable elements increases with iterations indicating that search in our engineered SimE



(a) SA: Total Power vs Iterations.

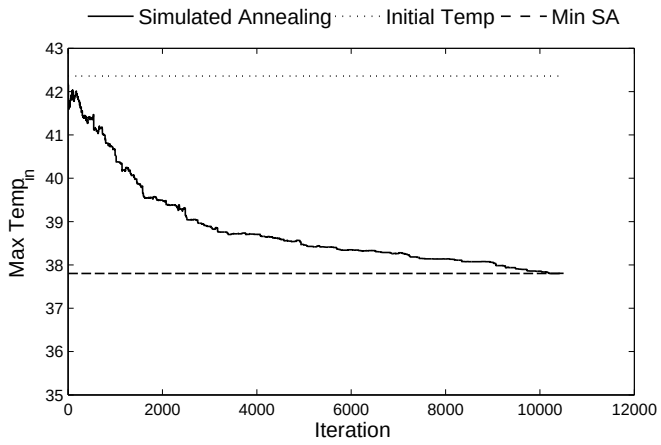


(b) TS: Total Power vs Iterations.

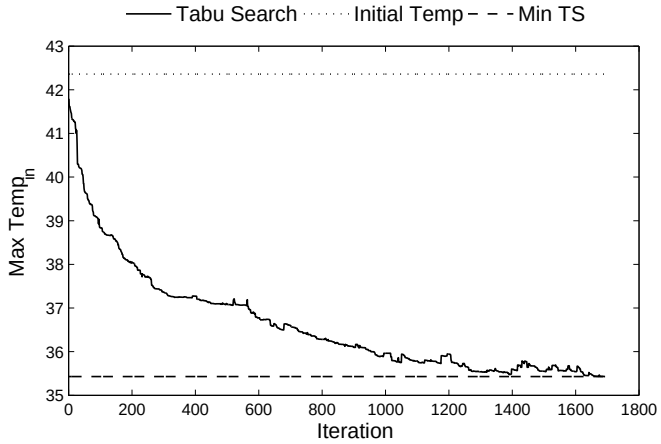


(c) SimE: Total Power vs Iterations.

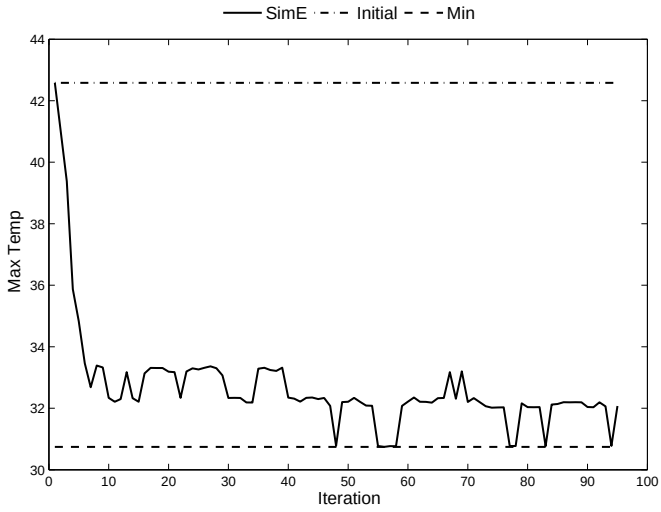
Figure 5.5: Change in Total Power consumption ($P_(total)$) (m) with iterations in (a) SA (b) TS (c) SimE.



(a) SA: Max Inlet Temperature vs Iterations.

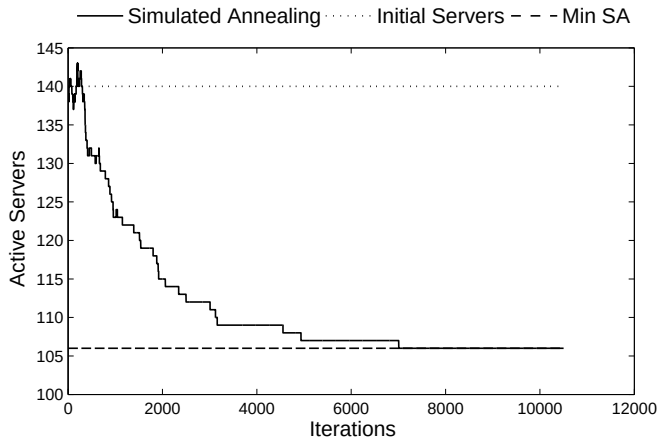


(b) TS: Max Inlet Temperature vs Iterations.

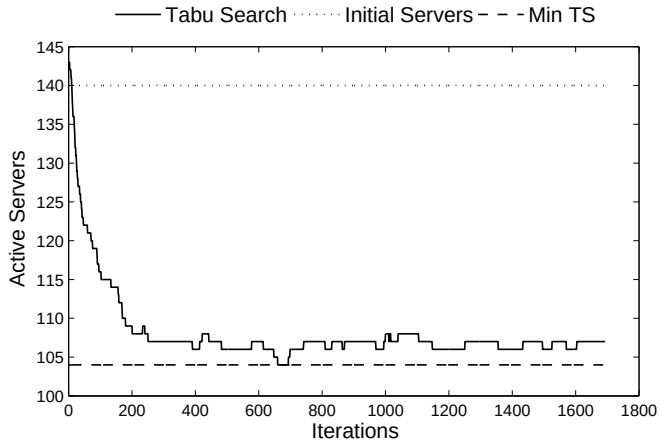


(c) SimE: Max Inlet Temperature vs Iterations.

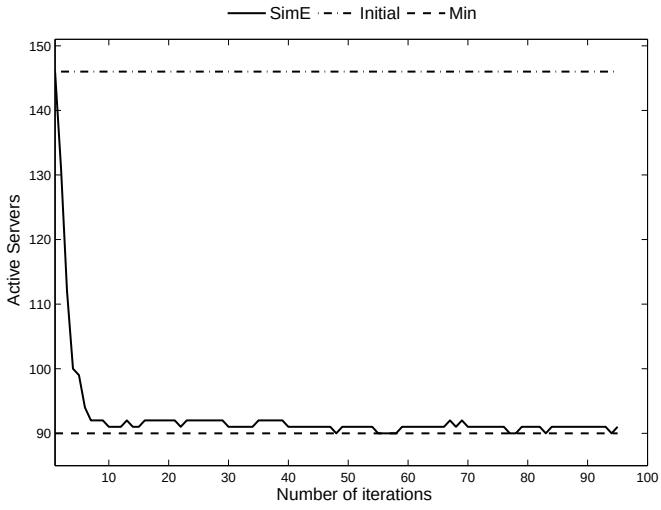
Figure 5.6: Change in max inlet temperature ($max(T_{in})$) with iterations in (a) SA (b) TS (c) SimE.



(a) SA: No. of Servers vs Iterations.



(b) TS: No. of Servers vs Iterations.



(c) SimE: No. of Servers vs Iterations.

Figure 5.7: Change in number of servers used (m) with iterations in (a) SA (b) TS (c) SimE.

implementation is progressing towards a solution where each VM is optimally assigned. Further indication of this is the reduction in the size of the selection set with iterations (Fig. 5.9).

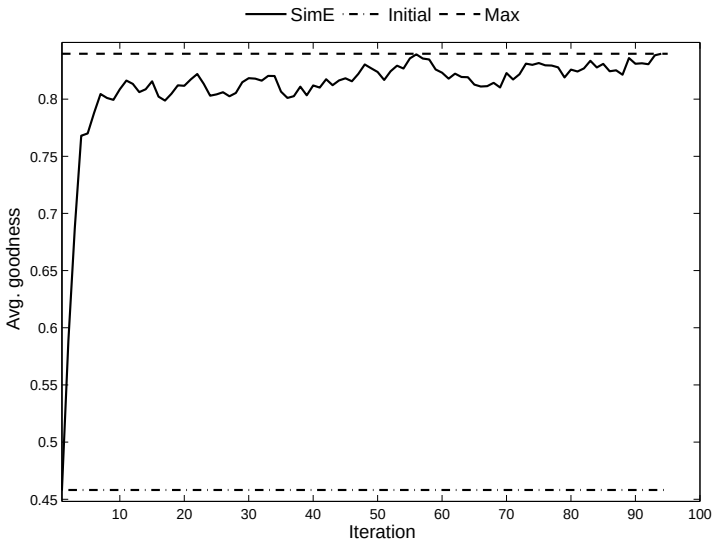


Figure 5.8: SimE: Change in the average goodness of VMs with iterations.

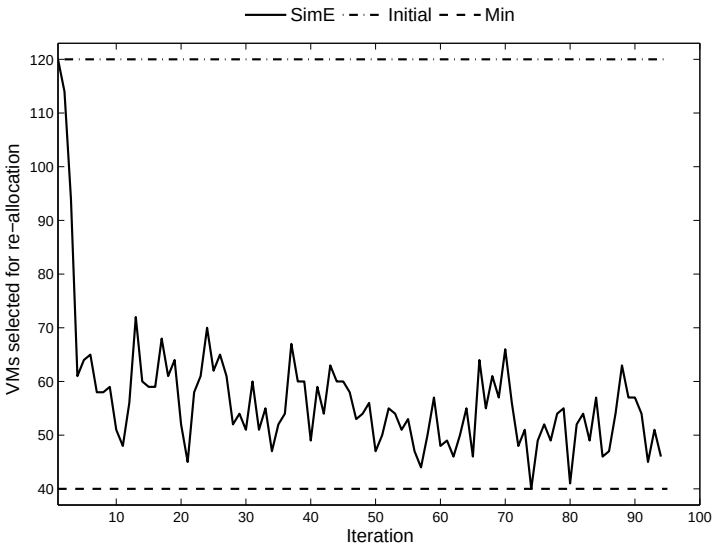


Figure 5.9: SimE: Change in the number of VMs selected for re-allocation with iterations.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

Computational power and cooling power are significant parts of a data center's operational costs. Previous studies attempt to minimize computational power and cooling power in an isolated manner without considering the interaction. In this work, we propose a joint optimization of cooling and computational power using evolutionary non-deterministic heuristics, Simulated Annealing, Tabu Search and Simulated Evolution. We evaluated its performance for a wide range of different problem instances. We showed that our approach is performing better as compared to those published in literature. It is also proved that performance of $SimE$, TS, and SA are independent of the correlation between different dimensions of VMs. This feature makes these heuristics desirable for all scenarios.

6.2 Future Work

In this work, we only considered the scenario where all the VM requests are known before placement and the controller allocates them at once, trying to find the optimal allocation in accordance with the objectives and constraints. Such situations arise when a data center starts its operation after a maintenance state or when the data center optimizer/ controller takes a decision at the back-end. However, in operational data centers VM requests arrive incrementally over time. In order to address this issue, it is recommended that future studies look into modifications of the algorithm that would work for an online scenario.

REFERENCES

[1] wikipedia.org. Cloud computing. [Online]. Available: http://en.wikipedia.org/wiki/Cloud_computing

[2] I. T. G. R. Team, in *Green IT : reality, benefits & best practices*. Ely, U.K. : IT Governance, 2008.

[3] G. eSustainability Initiative *et al.*, *SMART 2020: Enabling the low carbon economy in the information age*. Climate Group, 2008.

[4] D. J. Brown and C. Reams, "Toward energy-efficient computing," *Communications of the ACM*, vol. 53, no. 3, pp. 50–58, 2010.

[5] D. Filani, J. He, S. Gao, M. Rajappa, A. Kumar, P. Shah, and R. Nagappan, "Dynamic data center power management: Trends, issues, and solutions." *Intel Technology Journal*, vol. 12, no. 1, 2008.

[6] J. Hamilton, "Cooperative expendable micro-slice servers (cems): low cost, low power servers for internet-scale services," in *Conference on Innovative Data Systems Research (CIDR)*, 2009.

[7] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE computer*, vol. 40, no. 12, pp. 33–37, 2007.

[8] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The case for power management in web servers," in *Power aware computing*. Springer, 2002, pp. 261–289.

[9] R. Basmadjian, F. Niedermeier, and H. De Meer, "Modelling and analysing the power consumption of idle servers," in *Sustainable Internet and ICT for Sustainability (SustainIT)*. IEEE, 2012, pp. 1–9.

[10] X. Feng, R. Ge, and K. W. Cameron, "Power and energy profiling of scientific applications on distributed systems," in *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium*. 2005, pp. 34–34.

[11] Y. Fu, C. Lu, and H. Wang, "Robust control-theoretic thermal balancing for server clusters," in *IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*. 2010, pp. 1–11.

[12] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 103–116.

[13] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," in *Workshop on compilers and operating systems for low power*, vol. 180. Barcelona, Spain, 2001, pp. 182–195.

[14] L. Wang, G. von Laszewski, J. Dayal, and T. R. Furlani, "Thermal aware workload scheduling with backfilling for green data centers," in *Performance Computing and Communications Conference (IPCCC), IEEE 28th International*. 2009, pp. 289–296.

[15] C. Bash, C. Hyser, and C. Hoover, "Thermal policies and active workload migration within data centers," in *ASME InterPACK Conference collocated with the ASME Summer Heat Transfer Conference and the ASME 3rd International Conference on Energy Sustainability*. American Society of Mechanical Engineers, 2009, pp. 673–679.

[16] C. D. Patel, R. Sharma, C. E. Bash, and A. Beitelmal, "Thermal considerations in cooling large scale high compute density data centers," in *The Eighth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. IEEE, 2002, pp. 767–776.

[17] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, "Making scheduling "cool": Temperature-aware workload placement in data centers." in *USENIX annual technical conference, General Track*, 2005, pp. 61–75.

[18] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase, "Balance of power: Dynamic thermal management for internet data centers," *Internet Computing, IEEE*, vol. 9, no. 1, pp. 42–49, 2005.

[19] F. Ahmad and T. Vijaykumar, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *ACM Sigplan Notices*,

vol. 45, no. 3. ACM, 2010, pp. 243–256.

[20] Power regulator for proliant servers. [Online]. Available: http://h20000.www2.hp.com/bc/docs/support/SupportManual/c00300430/c0

[21] A. Gupta, R. Krishnaswamy, and K. Pruhs, "Online primal-dual for non-linear optimization with applications to speed scaling," in *Approximation and Online Algorithms*. Springer, 2013, pp. 173–186.

[22] M. Andrews, S. Antonakopoulos, and L. Zhang, "Minimum-cost network design with (dis)economies of scale," in *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Oct 2010, pp. 585–592.

[23] P. Mell and T. Grance, "The nist definition of cloud computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[24] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[25] Vmware esx server. [Online]. Available: http://www.vmware.com/products/esx

[26] Kernal based virtual machine. [Online]. Available: http://www.linux-kvm.org/page/MainPage

[27] Xensource inc, xen. [Online]. Available: http://www.xensource.com

[28] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[29] S. M. Sait and K. S. Shahid, "Engineering simulated evolution for virtual machine assignment problem," *Applied Intelligence*, pp. 1–12, 2015.

[30] S. M. Sait, A. Bala, and A. H. El-Maleh, "Cuckoo search based resource optimization of datacenters," *Applied Intelligence*
DOI: 10.1007/s10489-015-0710-x

[31] H. H. Kramer, V. Petrucci, A. Subramanian, and E. Uchoa, "A column generation approach for power-aware optimization of virtualized heterogeneous server clusters," *Computers & Industrial Engineering*, vol. 63, no. 3, pp. 652–662, 2012.

[32] S. K. Doddavula, M. Kaushik, and A. Jain, "Implementation of a fast vector packing algorithm and its application for server consolidation," in *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2011, pp. 332–339.

[33] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Int. CMG Conference*, 2007, pp. 399–406.

[34] R. F. Sullivan, "Alternating cold and hot aisles provides more reliable cooling for server farms," *Uptime Institute*, 2000.

[35] C. D. Patel, C. E. Bash, R. Sharma, M. Beitelmal, and R. Friedrich, "Smart cooling of data centers," in *ASME International Electronic Packaging Technical Conference and Exhibition*. American Society of Mechanical Engineers, 2003, pp. 129–137.

[36] T. Mukherjee, Q. Tang, C. Ziesman, S. K. Gupta, and P. Cayton, "Software architecture for dynamic thermal management in datacenters," in *2nd International Conference on Communication Systems Software and Middleware*. IEEE, 2007, pp. 1–11.

[37] A. M. Al-Qawasmeh, S. Pasricha, A. Maciejewski, H. J. Siegel *et al.*, "Power and thermal-aware workload allocation in heterogeneous data centers," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 477–491, 2015.

[38] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 13–23.

[39] E. Pakbaznia and M. Pedram, "Minimizing data center cooling and server power costs," in *Proceedings of the ACM/IEEE international symposium on Low power electronics and design*. ACM, 2009, pp. 145–150.

[40] Q. Tang, S. K. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458–1472, 2008.

[41] Q. Tang, T. Mukherjee, S. K. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Fourth International Conference on Intelligent Sensing and Information Processing(ICISIP)*. IEEE, 2006, pp. 203–208.

[42] S. Nahar, S. Sahni, and E. Shragowitz, "Simulated annealing and combinatorial optimization," *International Journal of Computer-Aided VLSI Design*, vol. 1, no. 1, pp. 1–23, 1989.

[43] S. M. Sait and H. Youssef, *Iterative computer algorithms with applications in engineering: solving combinatorial optimization problems*. IEEE Computer Society Press, 1999.

[44] M. Payne, P. Bristowe, and J. Joannopoulos, "Ab initio determination of the structure of a grain boundary by simulated quenching," *Physical review letters*, vol. 58, no. 13, p. 1348, 1987.

[45] L. Ingber, "Simulated annealing: Practice versus theory," *Mathematical and computer modelling*, vol. 18, no. 11, pp. 29–57, 1993.

[46] N. Metropolis *et al.*, "Equation of state calculations by fast computing machines," *Journal of Chem. Physics*, vol. 21, pp. 1087–1092, 1953.

[47] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[48] F. Glover, "Tabu searchpart ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[49] R.-M. Kling and P. Banerjee, "Esp: A new standard cell placement package using simulated evolution," in *Proceedings of the 24th ACM/IEEE Design Automation Conference*. ACM, 1987, pp. 60–66.

[50] S. M. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*. McGraw-Hill, Inc., 1994.

[51] L. Shi, J. Furlong, and R. Wang, "Empirical evaluation of vector bin packing algorithms for energy efficient data centers," in *IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2013, pp. 000 009–000 015.

[52] R. Panigrahy, K. Talwar, L. Uyeda, and U. Wieder, "Heuristics for vector bin packing," *research. microsoft. com*, 2011.

Vitae

- Name: Ali Raza
- Nationality: Pakistani
- Date of Birth: May 1, 1991
- Email: *aliraza.ece250@gmail.com*
- Permanent Address: Kot Addu, Pakistan

- Received Bachelor's degree in Electrical Engineering from University of Engineering & Technology (UET) Lahore, Pakistan in Sept 2012.
- Obtained Master's degree in Computer Engineering from King Fahd University of Petroleum & Minerals (KFUPM) Dhahran, Saudi Arabia in December 2015.